TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Professur für Softwaretechnologie

Bachelor-Arbeit

Abarbeitung und Speicherung von hochfrequenten Sensor-Daten in Smart Home Systemen

Manuel Krombholz

Geboren am: 22.07.1997 in Dresden, Sachsen
Matrikelnummer: 4604757

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

Betreuer
Dipl.-Inf. René Schöne

Betreuender Hochschullehrer
Prof. Dr. rer. nat habil. Uwe Aßmann

Eingereicht am: 24.08.2020

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit mit dem Titel *Abarbeitung und Speicherung von hochfrequenten Sensor-Daten in Smart Home Systemen* selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in der Arbeit angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Es waren keine weiteren Personen an der geistigen Herstellung der vorliegenden Arbeit beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Hochschulabschlusses führen kann.

Dresden, 24.08.2020

Manuel Krombholz

Zusammenfassung

Der sich stetig entwickelnde Markt intelligenter Smart Home Geräte und das wachsende Bedürfnis des Einsatzes solcher Geräte bedingen eine intensive wissenschaftliche Auseinandersetzung mit Smart Home Systemen. Es bedarf insbesondere wissenschaftlicher Ansätze, um relevante Komponenten für die Realisierung solcher Systeme entwickeln zu können. Die Herausforderung besteht vor allem in der Komplexität der Systeme. So bringt u. a. eine sich steigernde Anzahl miteinander verbundener Geräte auch wachsende Anforderungen im Hinblick auf die zu verarbeitende Datenrate mit sich. Diese Arbeit beschäftigt sich mit dem Problem der Performance von Smart Home Systemen auf softwaretechnologischer Ebene, insbesondere mit der Verarbeitung einer außerordentlich hohen Menge an Informationen und der daraus resultierenden Herausforderung im Hinblick auf die Speichereffizienz. Neben der Untersuchung von Arbeiten, die ebenfalls auf Lösungen für verwandte Probleme abzielen, wird im Rahmen der vorliegenden Arbeit zudem die Frequenz und Dauer der Event-Verarbeitung zweier Smart Home Systeme in Form eines Benchmarks gemessen. Dessen Ergebnisse werden nach einer Beschreibung der allgemeinen Funktionsweise beider Systeme erläutert. Zudem werden eigene Ansätze für ein Smart Home System und dessen Umgang mit den genannten Herausforderungen in Form eines Konzepts erarbeitet. Ein Proof-Of-Concept wird dargelegt, indem eine Implementation dieses Konzeptes evaluiert wird. Das erwähnte Benchmark-Werkzeug wird genutzt, um gleiche Metriken bei der implementierten Software zu messen. Die Ergebnisse werden denen der beiden anderen Systeme gegenübergestellt.

Inhaltsverzeichnis

Abkürzungen

CEP Complex Event Processing

JSON Javascript Object Notation

SHS Smart Home System

1 Einleitung

„Es liegt in der Natur des Menschen, Wege zu finden, die das tägliche Leben leichter und angenehmer [...] gestalten." (Zitat [3]) Wenig später fingen Wissenschaftler an, diese Vorstellungen teilweise in die Realität umzusetzen und banden Geräte in automatisierte Routinen ein. Heute beläuft sich die geschätzte Anzahl an Haushalten, welche mit Smart Home Geräten ausgestattet sind, auf 175 Millionen. Innerhalb der nächsten 4 Jahre soll sich diese Zahl verdoppeln.[46] Zudem sind auch Geräte, die andere Motive als die Maximierung des Komforts in den eigenen Wänden verfolgen, wie z. B. das Gefühl der Sicherheit zu steigern, in der Entwicklung und werden vermarktet. So bieten diverse Hersteller Fenster und Türschlösser an, welche sich mittels mobiler Endgeräte fernsteuern lassen.[50,49] Eine ähnliche Entwicklung zeigt die Variabilität an Umgebungen, in denen smarte Geräte Platz finden. Neben Wohnungen oder Wohnhäusern, stattet man mittlerweile auch Schiffe[15], Ferienhäuser[18] oder Hotels[25] mit diesen Technologien aus.
Mit der Vielfalt an Möglichkeiten in diesem informationstechnischen Bereich wird auch die Menge an einhergehenden Problemen größer, welche bei der zentralen Automatisierung mithilfe von Smart Home Systemen auftauchen. Diese bestehen sowohl im Bereich der Hardware als auch der Software. In dieser Arbeit soll eine Teilmenge dieser Probleme aus softwaretechnologischer Sicht genauer unter die Lupe genommen werden. Konkret handelt es sich um die Erhaltung einer akzeptablen Performance bei der Verarbeitung großer Mengen an Informationen sowie die Herausforderung, den verfügbaren persistenten Speicher dabei sinnvoll und schonend zu nutzen, um die enthaltenen Daten a posteriori weiterzuverarbeiten. Es wird untersucht, wie die Performance eines Smart Home Systems gemessen werden kann und Methodiken entwickelt, die dazu beitragen sollen, die Systeme den modernen Bedingungen und Umgebungen entsprechend anzupassen.

2 Grundlagen

Dieses Kapitel dient der Übermittlung grundlegenden und essentiellen Wissens der Themen, die in dieser Arbeit von Relevanz sind.

2.1 Smart Home

Für den Begriff „Smart Home“ (frei übersetzt: „Intelligentes Zuhause“) existiert keine einheitliche Definition, da die Thematik unterschiedlich aufgegriffen wird. Die TU Berlin definiert den Begriff wie folgt: „Ein Smart Home ist die Integration von Technologien und Dienstleistungen in das häusliche Umfeld, um die Lebensqualität, Sicherheit und Kommunikationsmöglichkeiten mit der Außenwelt zu verbessern.“ (übersetzt aus [52]) Eine andere Quelle versteht unter jenem Begriff hauptsächlich eine Umgebung, in der bislang manuell ausgeführte Abläufe digital automatisiert werden, wobei die dabei verwendeten Geräte zentral gesteuert werden.[59] Folglich stehen Technologien im Vordergrund, die Abläufe oder Aktionen in einer meist bewohnbaren Umgebung vereinfachen.

Smart Devices

Dabei werden die im Kontext von Smart Home vermarkteten Geräte oft ebenfalls mit den Termini smart oder intelligent beschrieben. Auch für diese Rubrik der Hardware ist keine allgemeingültige Definition bekannt. [45] bezeichnet jene Geräte als „mit Intelligenz ausgestattete Kleinstsysteme, die über das Internet of Things (IoT) vernetzt sind“. Zudem werden sie demnach für das Erfassen unterschiedlicher physikalischer Messgrößen verwendet, welche sie an zentrale Einheiten übermitteln. Auf der anderen Seite werden Smarte Geräte als physische Objekte betrachtet, welche über einen eingebetteten Prozessor, Speicher sowie eine Netzwerkverbindung verfügen und in der Lage sind, mittels einer Benutzer-Schnittstelle mit der physischen Umgebung zu kommunizieren.[56]

2.1.1 Nutzungsstatistik

Die Popularität von Smart Home Anwendungen nimmt stark zu. Wie in Abbildung 2.1 zu sehen ist, steigen die Kurven aller dort erwähnten Teilgebiete gleichmäßig an. Folglich ist diese Thematik in der modernen Welt von hoher Relevanz und bedarf Aufmerksamkeit bei der Entwicklung.

2.1.2 Einsatzorte

Neben dem Einsatz in Wohnungen und Wohnhäusern werden smarte Haushaltsgeräte mittlerweile auch in Umgebungen zweckähnlicher Natur eingesetzt. Dies können unter anderem Hotels und Schiffe wie z. B. Yachten oder Kreuzfahrschiffe sein.

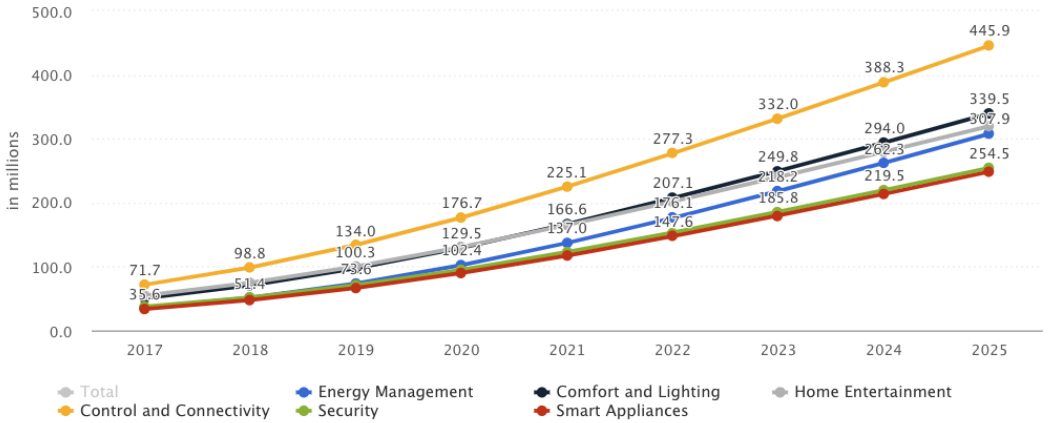

Abbildung 2.1: Statistik Nutzer von Smart Home [45]

2.2 Smart Home System

Smart Home Systeme sind Software-Produkte, welche den Vorteil bieten, komplette Einrichtungen wie eine Wohnung, ein Haus oder gar Grundstücke zentral und autonom zu steuern. Beispielsweise lassen sich Abläufe konfigurieren, die zu einem gewählten Ereignis automatisch gestartet werden. Ein Beispiel ist die Plattform zur Gebäudeautomatisierung OpenHAB. Grundlegende Bestandteile und Motive von Smart Home Systemen werden im Folgenden genauer dargelegt.

2.2.1 Funktionen

Hauptsächlich stellen Smart Home Systeme eine Schnittstelle für den Nutzer dar, um auf unterstützte und angemeldete Smart Home Geräte zuzugreifen oder diese anzusteuern. Es werden unterschiedliche Paletten an Funktionen angeboten.

Strukturieren

Mithilfe verfügbarer Konfigurationsmöglichkeiten lassen sich Geräte strukturieren und so etwa zu Gruppen zusammenfügen, um sie beispielsweise mit einer einfachen Interaktion wie z.B. einem Klick an- oder auszuschalten.[28,23]

Manuelles Ansteuern

Weiterhin existieren Produkte, die dem Nutzer zusätzlich eine grafische Benutzer-Oberfläche zur Verfügung stellen, um ihm eine einheitliche Übersicht über seine Geräte zu bieten und unter Umständen auch deren Zustände zu übermitteln oder ihn diese manuell ansteuern zu lassen.[12]

Automatisierung

Ein häufig gebotenes Feature ist die Automatisierung mithilfe von Regeln oder Skripten, welche vom Nutzer eingerichtet werden können. Im Fall von Regeln kann eine Reihe an Auslösern angegeben werden sowie Aktionen, die im Fall des Eintretens jener Auslöser gestartet werden sollen.[7,39]

2.2.2 Motive

Aufgrund der Vielzahl von Diensten und Anwendungen, welche in ein SHS integriert werden, lassen sich Anwendungsfelder unterschiedlich kategorisieren.[11] Im Folgenden werden mögliche Aspekte für den Einsatz von Smart Home aufgezählt und mit Beispielen veranschaulicht.

Sicherheitsaspekt

Der Einsatz von Geräten aus der Überwachungstechnik kann dazu beitragen, den bewohnten Raum sicherer zu gestalten. Dabei kann beispielsweise die Gefahr eines Diebstahls gemindert werden, indem Bewegungsmelder genutzt werden, um im Ernstfall einen Alarm auszulösen oder die Fenster zu entsprechenden Zeitpunkten geschlossen werden. Weiterhin können intelligente Lampen genutzt werden, um bei Abwesenheit ein bewohntes Haus zu simulieren.

Unterhaltungsaspekt

Der Unterhaltungsaspekt umfasst jede Form der Unterhaltung, die in einem Haus/einer Wohnung genossen werden kann. Beispielsweise kann eine Lichtszene automatisch an Ereignisse wie das Wiedergeben oder Pausieren eines Films sowie den Takt von Musik angepasst werden.

Komfortaspekt

Neben dem Aspekt der Unterhaltung lässt sich die Qualität des Wohnens auch auf passive Weise steigern. Mithilfe der entsprechenden Konfiguration im Automatisierungsprozess können Effekte wie das Blenden durch die Sonne vermieden werden, indem unterstützte Jalousien o.ä. so angesteuert werden, um die Räumlichkeiten automatisch abzudunkeln. Zudem ist es möglich, gewünschte Innentemperaturen mittels eingerichteter Thermostate konstant zu halten. Smarte Heizungen, Fenster und Klimaanlagen können dabei eine essentielle Rolle spielen.

Umweltaspekt

Auch die Umwelt kann durch die gezielte Konfiguration eines Smart Homes geschont werden. Eine Automatisierung von ansteuerbaren Fenstern und diversen Temperatursensoren ermöglicht es z.B., dass kalte Außentemperaturen genutzt werden, um die Innentemperatur zu senken anstatt für diesen Zweck die Klimaanlage zu verwenden. Bei zu hohen Außentemperaturen hingegen können die Fenster geschlossen werden, um die Effizienz der Klimaanlage zu erhöhen. Auch die Energie, welche durch das Verwenden von smarten Lampen bei Abwesenheit des Nutzers verloren geht, kann gespart werden, indem Lampen automatisch abgeschaltet werden.

Gesundheitsaspekt

Im Bereich Gesundheit kann Smart Home ebenfalls einen Beitrag leisten. Mittels Sensorik können gefährliche Situationen wie beispielsweise undichte Quellen von Gasen vorzeitig erkannt werden. Im Extremfall können bei Bedarf zudem automatisch Angehörige oder Ärzte informiert werden. Vorteilhaft ist letzteres auch, falls entsprechende Anlagen zur Erkennung von Stürzen eines Menschen installiert sind.[47,31]

Finanzieller Aspekt

Auch für das Aufzeichnen von Verbrauchsständen der Energie können smarte Geräte von Relevanz sein. Neben der Berechnung des Gesamtverbrauchs können Geräte mit hohem Verbrauch entdeckt werden, um finanzielle Kosten einzuschränken.

2.2.3 Eventverarbeitung

Der Automatisierung von SHS liegt meist die Verarbeitung von Events zugrunde.

Complex Event Processing (CEP)

Unter dem Begriff Complex Event Processing werden Techniken und Methoden zusammengefasst, welche darauf spezialisiert sind, Ereignisse in Echtzeit kontinuierlich zu verarbeiten. Ziel ist es, komplexe Ereignisse aus eingehenden Informationen abzuleiten. So können u. a. hohe Datenraten, die auf Instanzen der hauptsächlichen Eventverarbeitung wirken, verringert und Ressourcen gespart werden. Ein Anwendungsgebiet von CEP sind Sensor-Netzwerke. In diesen werden Messdaten der Außenwelt an Systeme, die der Datensammlung dienen, übermittelt. Fehler können dabei minimiert werden, indem beispielsweise Daten mehrerer Sensoren zusammengefasst werden.[20] Um hohe Datenraten innerhalb eines Smart Home Systems in Echtzeit zu verarbeiten, kann CEP im Designprozess des Systems hilfreich sein. Im Kapitel 5 werden dazu einige Ansätze beschrieben.

2.2.4 MQTT

MQTT ist ein weit verbreitetes Protokoll, welches u. a. bei der Kommunikation von IoT-Geräten und -Anwendungen zum Einsatz kommt. Es basiert auf dem Publish-Subscribe-Verfahren und nutzt das Transmission Control Protocol (TCP). Nachrichten sind dabei meist klein und werden geordnet und verlustfrei bidirektional übertragen. Das Protokoll wird innerhalb des OSI-Schichten-Modells in die Anwendungsschicht eingeordnet.[14] Um Nachrichten zu übermitteln, wird ein MQTT-Broker benötigt, welcher das Zentrum des Austauschs darstellt. Nachdem Clients die gewünschte(n) Topic(s) abonniert und somit an den Broker übermittelt haben, verteilt dieser empfangene Nachrichten, welche mit dem entsprechenden Topic versehen sind, an dessen Abonnierenden.[34] Aufgrund des leichten Gewichts ist das Protokoll beliebt bei der Kommunikation von Smart Home Komponenten. Vor allem bei Sensoren wie gewöhnlichen Thermometern oder Bewegungsmeldern, deren ausgehende Nachrichten meist nur wenige Bytes und keine Medien wie Bilder oder Videos enthalten, ist die Nutzung von MQTT vorteilhaft.

2.2.5 Probleme und Herausforderungen

Hinsichtlich der Nutzung und Entwicklung von Smart Home Systemen ergeben sich mehrere Risiken und Herausforderungen. Einige davon werden mit zunehmendem Einfluss des Systems in das tägliche Leben größer. Diese Dinge werden nun aufgezeigt.

Sicherheit

Gewöhnlich werden Geräte wie Lampen oder Heizungen aber auch sicherheitskritische Elemente wie Fenster und Türen manuell bedient oder geschalten. Dazu muss die agierende Person physisch in der Nähe des Gerätes sein und sich somit im Haus befinden (oder im Fall der Tür einen Türschlüssel besitzen). Durch den Einsatz eines Smart Home Systems bleibt diese Sicherheitsschranke aus. Für potentielle Einbrecher hat dies den Vorteil, dass ein Zugriff aus der Ferne möglich ist, falls in die richtigen Netzwerke und Umgebungen eingebrochen wurde. Durch den Bekanntheitsgrad verbreiteter Smart Home Systeme werden regelmäßig neue Schwachstellen aufgedeckt und veröffentlicht. Danach ist der Aufwand, mehrere Geräte zu manipulieren, geringer als ohne ein solches System, da einheitliche Oberflächen mit Übersichten und Schnittstellen mit Möglichkeiten der Steuerung im Fall des Einbruchs ausgenutzt werden können. Ausreichend ist dabei das Kompromittieren von intelligenten Sprachassistenten, welche in der Lage sind, eingerichtete Geräte direkt oder indirekt anzusteuern. Diverse Quellen[35,2] berichten, wie sie dies mit einfachen Mitteln bewerkstelligen konnten.

Performance

Mit steigender Bewohnerzahl oder Wohnungsgröße kann sich die Anzahl an installierten smarten Komponenten fortlaufend erhöhen. Damit geht eine Steigerung der Datenrate, welche auf das Sy-

stem wirkt, einher. Eingehende Daten müssen unter Umständen in Echtzeit verarbeitet werden. Das Problem Performance wird in dieser Arbeit besonders beleuchtet.

Schnittstellen

Trotz vorhandener Protokolle zur Übertragung von Steuerbefehlen und Zustandsänderungen tendieren Gerätehersteller meist zur Kreation einer eigenen Schnittstelle bzw. einem eigenen Protokoll. Dies führt dazu, dass Smart Home Systeme nicht unmittelbar nach der Basis-Installation mit diesen Geräten kommunizieren können und stattdessen eine Art Treiber in das System integriert werden muss, um diesem geräte-spezifische sowie allgegenwertige Zustände (wie An/Aus) verständlich zu übermitteln oder entgegenzunehmen. Aus diesem Grund wird derzeit an der Entwicklung eines neuen Verbindungsstandards gearbeitet.[5] Das Projekt mit Namen „Connected Home over IP“, welches maßgeblich von Unternehmen wie Amazon, Apple und Google vorangetrieben wird, soll die Kompatibilität zwischen Smart Home Produkten steigern sowie einen hohen Grad an Sicherheit gewährleisten. Dazu werden IP-basierte Netzwerktechnologien definiert, um Geräte zu zertifizieren. Ein solcher Standard könnte eine internationale Lösung für die nahtlose Verknüpfung unterschiedlichster Smart Home Elemente sein, ohne dabei separate Adapter zu entwickeln.

3 Die Systeme OpenHAB und Eraser

Für die Automatisierung und Verwaltung von Geräten in einem Haushalt existieren bereits zahlreiche Lösungen.[32,27,19,24]
In diesem Kapitel werden zwei solcher Produkte vorgestellt und anschließend deren Performance anhand eines Benchmarks gemessen. Selektiert wurden dazu die in Java entwickelte Lösung OpenHAB sowie die am Lehrstuhl der TU Dresden entstandene Software Eraser.

3.1 Einführung

Zunächst werden Informationen über die Smart Home Systeme als solche näher gebracht.

3.1.1 OpenHAB

OpenHAB ist eine Software-basierte Lösung zur Einrichtung und Steuerung eines Systems, mit deren Hilfe IoT-Geräte (klassischerweise Geräte, welche in Wohnungen und Räumen angebracht sind) passend und automatisiert angesteuert werden können. Der Begriff HAB leitet sich von dem englischen Begriff "Home Automation Bus" ab. Ziel der Software ist es, dem Nutzer eine benutzerfreundliche Steuerung und Automatisierung seiner Geräte zur Verfügung zu stellen. Mit dieser kann er zeit- und aufwandssparend häufig wiederkehrende Abläufe von Ansteuerungen selbst auslösen. Weiterhin bietet OpenHAB mehrere Oberflächen zur Verwaltung an. Diese können mit einem Browser aufgerufen und genutzt werden.

Funktionsweise

Die Smart Home Lösung OpenHAB basiert grundlegend auf der Softwareplattform OSGi. Darüber ist eine modulare Struktur von Kernkomponenten der Software verbaut, welche mit den jeweiligen Erweiterungen wie der Logik für die Automatisierung oder der Benutzeroberfläche interagiert. Abbildung 3.1 zeigt die Architektur des Systems.

Benutzerdefinierte Einheiten

Um die Konfiguration von Geräten, Automatisierungen sowie Oberflächen zur Übersicht nach eigenen Wünschen und Vorstellungen vornehmen zu können, bieten sich innerhalb OpenHAB diverse Entitäten an. Sie können mithilfe von Dateien eingerichtet und angepasst werden. Geräte werden im Kontext von OpenHAB Things (Dinge) genannt. Ihnen können ein Anzeigename sowie eine einmalige ID zugewiesen werden. Da der Zustand eines Gerätes in mehrere Teile (vergleichbar mit Attributen) untergliedert werden kann, ist diese Struktur auch in die Things von OpenHAB integriert und wird

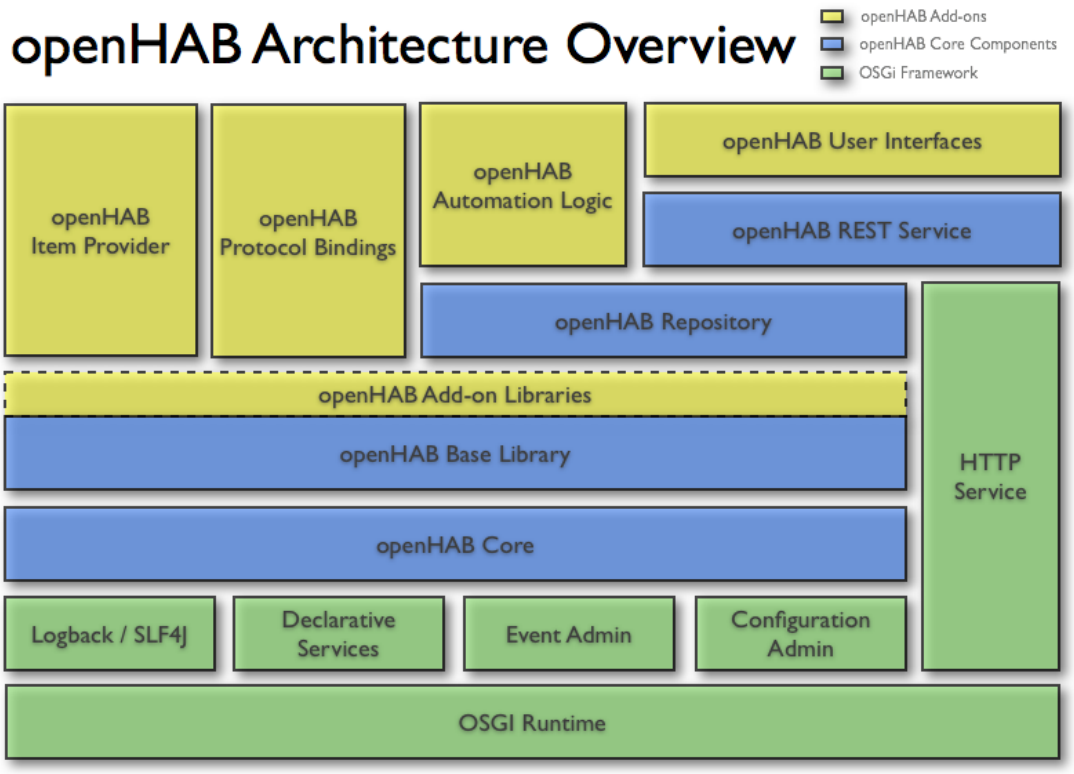

Abbildung 3.1: Architektur von OpenHAB [37]

als Channel (Kanal) bezeichnet. Channels erfüllen den Zweck, als Kanal die entsprechenden Attribut-Werte entweder an OpenHAB oder das Gerät selbst zu übermitteln.[54]
Items dienen der Ansteuerung von Geräten. Sie bestehen aus einer ID sowie einem Datentyp und können an einen oder mehrere Channels von Geräten gebunden werden. Das bedeutet, dass die Software den Zustand des Items mit dem Zustand jedes angebundenen Channels synchronisiert. Weiterhin besteht die Möglichkeit, Items auf Oberflächen in Form einer Kachel darzustellen. Damit wird ein Schnellzugriff für das Auslesen des aktuellen Zustands sowie die Funktion zum Ansteuern bereitgestellt.[28]
Mithilfe von Rules (Regeln) lassen sich diverse Automationsroutinen für konfigurierte Things einrichten. Solche Regeln bestehen neben der ID aus zwei Teilen. Dies sind die Auslösebedingung und der Skriptblock. Für letzteres kann unter anderem auch eine imperative Sprache verwendet werden.[39]
Bindings (Bindungen) verfolgen den Zweck, zwischen OpenHAB selbst und den unterstützen Geräten des jeweiligen Bindings zu kommunizieren. Folglich lassen sie sich mit Treibern vergleichen. Diese Entitäten lassen sich entweder manuell als Datei anlegen oder aus einem speziellen Store beziehen.[36]

Nachteile

Innerhalb des OpenHAB besteht nicht die Möglichkeit, mehrere Benutzer zu verwalten und deren Aktionen mithilfe eines Rechtesystems zu überprüfen. So hat jeder Anwender im Netzwerk Zugriff auf alle Funktionen, Geräte und Mechanismen des gesamten Systems. Für eine Zugriffskontrolle bedarf es weiterer Programmierarbeit.[17] Lediglich IP-Adressen lassen sich filtern, jedoch kann dadurch nur der vollständige Zugriff verweigert werden, ohne dabei genau zwischen Funktionalitäten zu differenzieren wie bei einer Zugriffskontrolle.[43]

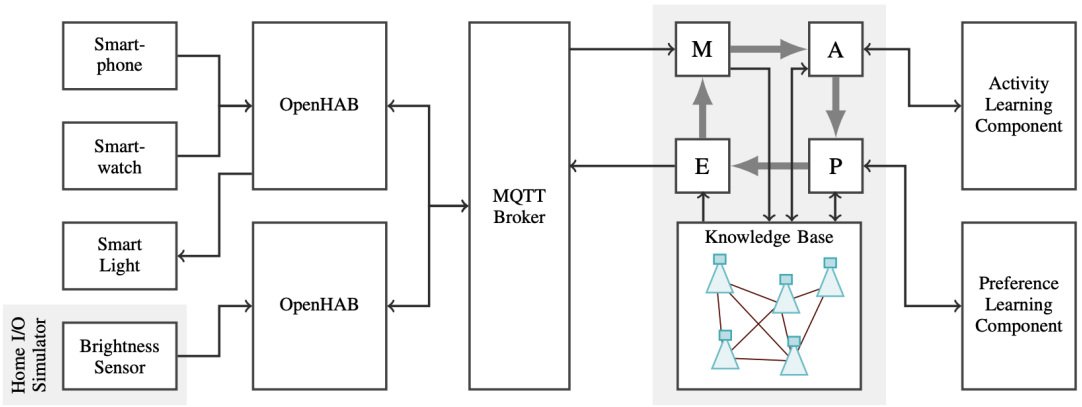

Abbildung 3.2: Architektur von Eraser [42]

3.1.2 Eraser

Im Rahmen des Softwaretechnologie Lehrstuhls der TU Dresden wurde Eraser von dem wissenschaftlichen Mitarbeiter und Betreuer dieser Arbeit René Schöne entwickelt. Neben Funktionen für die Automatisierung bietet die Software die Möglichkeit, bestehende Smart Home Systeme zu erweitern. Der Mehrwert liegt dabei in der Integration von Algorithmen für maschinelles Lernen in die zu erweiternden Systeme. Diese können modular vom Nutzer in Eraser eingebracht werden. Dadurch ist es beispielsweise möglich, mithilfe von neuronalen Netzen die Vorlieben eines Benutzers beim Umgang mit der Software zu lernen und ihn somit dabei zu unterstützen, die Umgebung zu automatisieren.[42]

Funktionsweise

Umgesetzt wurde die Lösung mithilfe von relationalen Referenzattribut-Grammatiken. Diese stellen eine Erweiterung zu kontextfreien Grammatiken dar. Die Architektur der Software baut auf einer MAPE-Architektur auf. Darin enthalten ist ein Regelkreis, welcher aus vier Teilen besteht, um mit erhaltenen Informationen dynamisch umzugehen. In diesem werden Informationen zuerst aufgenommen (Monitor) und anschließend analysiert, um daraus entsprechende Schlussfolgerungen zu ziehen (Analyze). Schließlich werden Aktionen aufbereitet, um eine gewünschte Anpassung vorzunehmen (Plan), wobei diese im Anschluss ausgeführt werden (Execute).[57] In Abbildung 3.2 ist die Architektur von Eraser und die Anbindung an die zuvor vorgestellte Software OpenHAB dargestellt. Dazu muss ein spezielles Binding in OpenHAB integriert sowie die Konfiguration für OpenHAB in Eraser eingerichtet werden. Dies ermöglicht die Kommunikation zwischen den beiden Systemen, so dass Eraser Einfluss auf die Steuerung angemeldete Geräte seitens OpenHAB nehmen sowie deren Zustandsänderungen erhalten und verarbeiten kann.

Benutzerdefinierte Einheiten

In der entsprechenden Konfigurationsdatei können die benutzerdefinierten Einheiten definiert werden. Die Bezeichnungen der Einheiten kann mit denen von OpenHAB verglichen werden. Physische Geräte werden innerhalb Eraser ebenfalls als Things bezeichnet und bestehen aus Channels, welche die Attribute dieser Geräte repräsentieren wie z. B. die Helligkeit einer Lampe. Mithilfe von Links können diese an Items gebunden werden. Items selbst verfügen über einen Datentyp und können durch Groups strukturiert werden.

3.2 Benchmark

Im Folgenden sollen die Grenzen der Performance von beiden vorgestellten Lösungen OpenHAB und Eraser anhand von Benchmark-Szenarios aufgezeigt werden. Dazu wird eine eigens geschriebene Software verwendet.
Der Fokus soll dabei auf der Funktionalität eines Messvorgangs von Lese-, Verarbeitungs- und Speicherzeit als Gesamtzeit und der Berechnung der Verarbeitungsfrequenz liegen. Um die Grenzen der Testobjekte zu erfahren, werden die einzelnen Systeme dafür mit hochfrequenten Mess-Daten, welche von simulierten Geräten ausgehen, unter Last gestellt. Diese Daten sollen in den Systemen Zustandsänderungen angemeldeter Geräte simulieren. Für die Versuche werden die jeweiligen Reaktionen der Systeme aufgezeichnet und anschließend eine durchschnittliche Verarbeitungszeit sowie eine Verarbeitungsfrequenz berechnet. Neben der Frequenz der Datenpakete wird ebenso die Anzahl der Sensoren systematisch erhöht. Die Gesamtsendefrequenz des Sendevorgangs entspricht in jedem Durchlauf der Anzahl der Sensoren multipliziert mit der aktuellen Frequenz der Datenpakete eines jeden Gerätes. Die Ergebnisse des Benchmarks werden im Kapitel 8 erneut aufgegriffen und dort mit den Testergebnissen einer eigenen Implementation eines Smart Home Systems verglichen.

3.2.1 Technische Spezifikationen

Die Durchführung des Benchmarks erfordert diverse Komponenten an Hardware und Software sowie Umgebungen, welche im Folgenden kurz aufgeführt werden.

Hardware

Ein Raspberry Pi 4 Model B dient bei den Versuchen als Server. Der Computer verfügt über folgende Ausstattung:

- RAM: 4 GB LPDDR4
- Prozessor: Broadcom BCM2711, Quad core Cortex-A72 64-bit SoC @ 1.5 GHz

Der Benchmark selbst wird von einem MacBook Air Model 2015 gesteuert. Der Computer verfügt über folgende Ausstattung:

- RAM: 8 GB DDR3
- Prozessor: 2,2 GHz Dual-Core Intel Core i7

Software

Der Open-Source Message Broker „mosquitto“ wird bei dem Versuch als MQTT-Broker verwendet und ist neben OpenHAB und Eraser auf dem Server installiert und eingerichtet.

3.2.2 Konfiguration und Vorbereitung

OpenHAB

Um OpenHAB die Kommunikation zwischen der Software und den generischen Geräten mittels MQTT zu ermöglichen, werden zuerst die Erweiterungen „MQTT Binding“ und „JSONPath Transformation“ installiert. Die Adresse des Brokers wird dem System danach mitgeteilt. Nun werden benötigte Geräte als Things nach dem von OpenHAB vorgesehenen Format beschrieben. Die Geräte beinhalten jeweils einen Channel und sind dadurch mit simplen Thermometern vergleichbar. Jeder Channel kann durch den Versand einer MQTT-Nachricht unter entsprechendem Topic (welches aus dem Schlüsselwort „benchmarkSend“ und dem jeweiligen Gerätenamen getrennt durch ein „/“ besteht) bearbeitet werden. Um die Konfiguration innerhalb OpenHAB möglichst realitätsnah zu

gestalten, wird jeder der Channels an ein dediziertes Item gebunden. Der Zustand eines Channels wird somit mit seinem zugehörigen Item synchronisiert. Pro Item wird eine Regel (innerhalb OpenHAB Rule genannt) angelegt, die bei einer Änderung des Item-Zustandes eine MQTT-Nachricht seitens OpenHAB ausgibt. Diese Botschaft setzt sich aus dem Schlüsselwort „benchmarkRec“ und dem Gerätenamen zusammen. Beides wird erneut durch ein „/“ getrennt. Die drei Artefakte (Things, Items, Rules) werden am Ende der Vorbereitung in den vorgesehenen Ordnern von OpenHAB abgelegt.

Eraser

Um Geräte bei Eraser anzulegen, werden diese zeilenweise als Items in die dafür vorgesehene Datei geschrieben. Dabei wird ein Topic angegeben, um das Gerät anzusprechen und eines, um seine ausgegebenen Informationen zu erhalten. Für das Simulieren einer Regel wird pro Gerät A ein weiteres Gerät B angelegt. In jedem Gerät A wird ein Attribut hinzugefügt, welches ermöglicht, eingehende Zustandsänderungen des Gerätes A auf das Gerät B zu übertragen. Wenn Gerät A folglich eine Zustandsänderung mitteilt, wird diese von Eraser auf das Gerät B übertragen und aufgrund des konfigurierten Topics in Gerät B via MQTT veröffentlicht.

3.2.3 Ablauf des Benchmarks

Nach der Einrichtung beginnt das Benchmark-Werkzeug mit einer festgelegten Geräteanzahl und Datenfrequenz für die Benchmark-Konfiguration des initialen Durchlaufs. Anschließend simuliert das Werkzeug in entsprechender Frequenz drei Sekunden lang Zustandsänderungen bei einer Menge von Geräten. Sowohl die Frequenz als auch die Geräteanzahl wird aus der Benchmark-Konfiguration entnommen. In den dabei versandten Nachrichten befindet sich ein genauer Zeitstempel. Dieser wird später für die Berechnung der Übertragungszeit benötigt, wenn das System reagiert und die entsprechenden Zustandsänderungen an das Werkzeug zurück übermittelt. Um den Systemen die Unterscheidung der simulierten Geräte zu ermöglichen, werden die Botschaften unter dem gerätespezifischen Topic versandt. Das Werkzeug wartet anschließend, bis die konfigurierten Regeln auf die Änderungen der Gerätezustände reagieren und lauscht mittels MQTT auf dem Topic „benchmarkRec“, bis alle Änderungen des aktuellen Durchlaufs empfangen wurden. Entspricht das Ergebnis dem Erwartungsbild, (d. h. die Anzahl der gemeldeten Zustandsänderungen ist korrekt) werden entsprechende Metriken berechnet und ein neuer Durchlauf mit höherer Datenfrequenz bzw. Geräteanzahl kann starten.

3.2.4 Maxima vollständiger Datensätze

Zunächst sollen maximale Konfigurationen gefunden werden, welche eine Grenze zu dem Bereich bilden, in dem die Systeme keine vollständigen Datensätze mehr übermitteln können. Ein Datensatz ist vollständig, wenn die Anzahl der gesendeten Werte der Anzahl der empfangenen entspricht. In Versuch 1 wird eine maximale Konfiguration gesucht, bei der die Anzahl der Geräte und deren jeweilige Datenrate gleich sind. Versuch 2 besteht darin, eine maximale Konfiguration mit nur einem Gerät zu finden. Versuch 3 hingegen bezweckt das Ersuchen einer Konfiguration mit einer Datenrate von einer Änderung pro Sekunde und Gerät.

System	Versuch 1	Versuch 2	Versuch 3
OpenHAB	12G, 12Ä	1G, 20Ä	350G, 1Ä
Eraser	Kein Maximum	Kein Maximum	Kein Maximum

Folgende Tabellen zeigen für diese und alle weiteren angewandten Benchmark-Konfigurationen, ob das System den Datensatz vollständig übermittelt hat. Dabei bildet sich das Tupel der Konfiguration aus der Geräteanzahl und der Änderungsfrequenz pro Gerät

System	(1,20)	(11,11)	(350,1)	(30,30)	(1500,1)
OpenHAB	X	X	X		
Eraser	X	X	X	X	X

System	(45,45)	(2200,1)	(4000,1)	(5000,1)	(1,5500)	(80,80)	(120,120)
OpenHAB							
Eraser	X	X	X	X	X	X	X

Hingegen ist die maximale theoretische Gesamtsendefrequenz bei OpenHAB deutlich geringer als die bei Eraser. Ab einer Benchmark-Konfiguration von 49 Geräten und 64 Änderungen pro Sekunde und Gerät kann OpenHAB nach wenigen Sekunden gar keine Regel mehr korrekt ausführen, da es vermutlich nicht mehr im Stande ist, die hohe Menge an Nachrichten mittels MQTT zu veröffentlichen. Diese Information wurde aus dem Log interpretiert. Für Konfigurationen mit 4000 Geräten und einer Änderung pro Sekunde und Gerät konnte OpenHAB nicht ordnungsgemäß die Regeln verarbeiten, obwohl diese wohlgeformt waren. Für das System Eraser wurde kein Maximum gefunden. Auch innerhalb Benchmarks mit Konfigurationen bis zu 120 Geräten sowie 120 Änderungen pro Sekunde, entsprachen die erhaltenen Daten immer noch dem Erwartungsbild. Messergebnisse bei höheren Konfigurationen erweisen sich jedoch nicht als aussagekräftig, da die effektive Gesamtsendefrequenz[1] nicht weiter erhöht werden konnte und die Verarbeitungsfrequenz der Systeme ebenfalls nicht. Daher soll diese Konfiguration die Grenze aller kommenden Versuche sein. Außerdem wurde festgestellt, dass das System Eraser nach mehreren Versuchen mit einer effektiven Gesamtsendefrequenz von 2400 Hz den verfügbaren Heap überschreitet und in einen unbrauchbaren Zustand übergeht. („java.lang.OutOfMemoryError: Java heap space“). Diese Information wurde ebenfalls aus dem Log entnommen.

3.2.5 Geschwindigkeit

Nun werden die beiden Systeme mit den gefundenen Maxima in Kombination mit weiteren erdachten Konfigurationen erneut unter Last gesetzt. Dabei wird jeder Benchmark fünf Mal ausgeführt, um Abweichungen mit einzubeziehen. In der Messung werden nun die durchschnittliche Verarbeitungszeit einer Zustandsänderung sowie die Frequenz der Verarbeitung notiert. Die beiden Box-Plots zeigen diese Werte nach System gruppiert für die einzelnen Benchmark-Konfigurationen an, welche nach deren theoretischer Gesamtfrequenz sortiert sind. Dabei sind die Ergebnisse für unvollständige Datensätze im Benchmark inbegriffen. Aus der Last-Grenze von OpenHAB sowie den Diagrammen in den Abbildungen 3.3 und 3.4 kann entnommen werden, dass die Performance Eventverarbeitung von OpenHAB und Eraser weniger von der theoretischen Gesamtsendefrequenz abhängt als von der Geräteanzahl. Vor allem bei höheren theoretischen Gesamtsendefrequenzen erreicht Eraser bessere Ergebnisse.

[1] Die effektive Gesamtsendefrequenz fällt meist geringer als die theoretische Gesamtsendefrequenz aus.

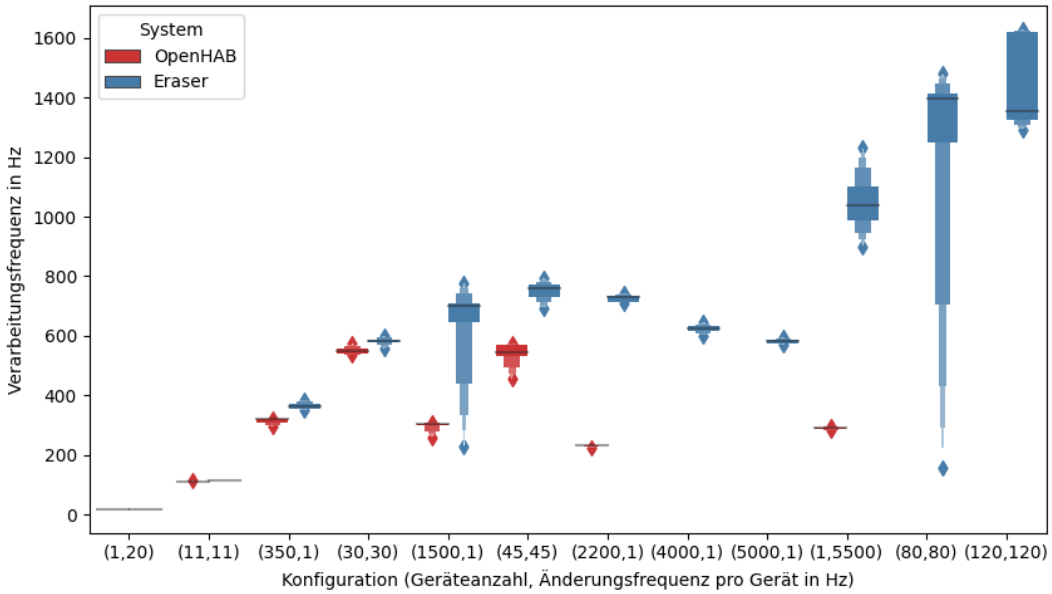

Abbildung 3.3: Verarbeitungsfrequenzen OpenHAB und Eraser

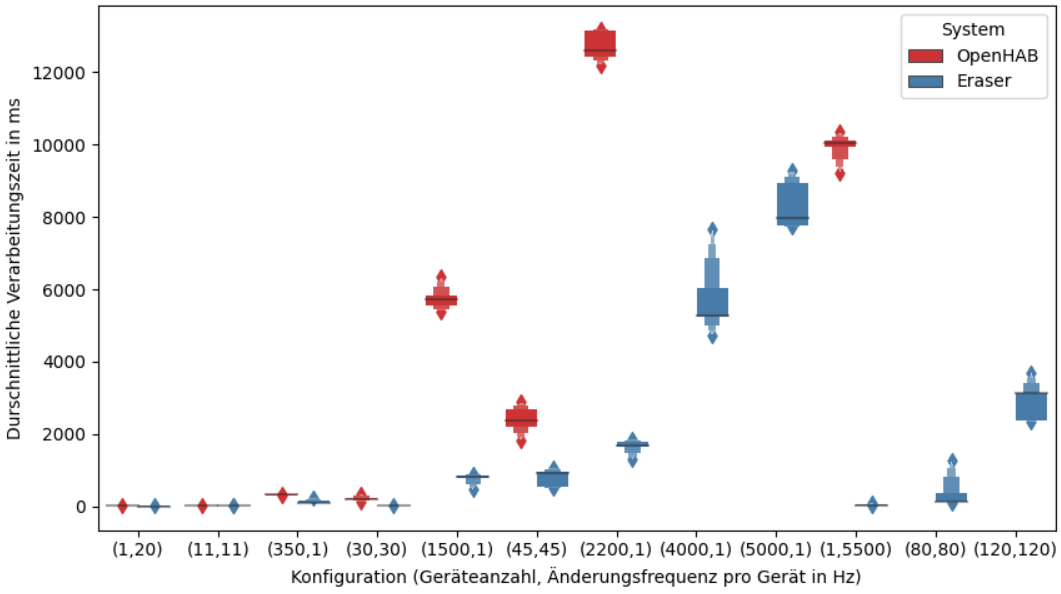

Abbildung 3.4: Verarbeitungszeiten OpenHAB und Eraser

4 Problem

Wie bereits erläutert, nimmt die Vielfalt an smarten Haushaltsgeräten am Markt zu. Damit wachsen die Einsatzmöglichkeiten sowie die Varianz an räumlichen Umgebungen, in denen jene Geräte verwendet werden.[48] Daraus resultiert eine breitere Zielgruppe für Smart Home Produkte, was sich letztlich auch auf die Anzahl der verbundenen Geräte pro Haushalt bzw. pro Einrichtung auswirkt.[1] Neben der Erweiterung der Gerätevielfalt werden auch die Einrichtungen variabler. Domänen wie Schiffe, Ferienwohnungen oder Hotels profitieren mittlerweile ebenfalls von der technischen Errungenschaft. Dadurch ergeben sich neue bzw. erschwerte Bedingungen für die reibungslose Funktionstüchtigkeit von Smart Home Systemen. In diesem Kapitel sollen die Herausforderungen und Probleme, welche beim Entwickeln solcher Produkte entstehen, sowie deren Einflussfaktoren dargelegt werden.

4.1 Signifikanz der Datenrate

Das Management mehrerer Geräte hat zur Folge, dass hierbei eine nicht unerhebliche Datenrate auf das SHS wirkt. Jene Rate resultiert aus Datenpaketen, welche in das System eingespeist werden, um Zustandsänderungen zu übermitteln. Diese können anschließend für weitere Steuerprozesse wie die Eventverarbeitung genutzt bzw. für späteren Zugriff persistent abgelegt werden.

Zudem wird die Datenrate erhöht, indem zusätzlich ausgehender Datenverkehr durch die Reaktion auf eingehende Informationen entsteht, falls die Automatisierung entsprechend konfiguriert ist. Die Rate hängt folglich auch von der Anzahl der verbundenen Geräte ab. Je mehr solcher Komponenten gleichzeitig Datenpakete an das System senden oder davon empfangen, desto größer ist die Frequenz der Rate und desto höher die erforderliche Performance der Lösung. Im Folgenden soll untersucht werden, welche weiteren Faktoren einer Einrichtung, die durch ein SHS verwaltet wird, einen Einfluss auf die Geräteanzahl oder die eingehende Datenrate selbst haben.

Flächengröße der Einrichtung

Wird Überwachungstechnik zur Erhöhung der Sicherheit eines Geländes oder von Räumlichkeiten eingesetzt, müssen eine Abdeckung der gesamten Fläche organisiert und tote Winkel vermieden werden. Somit ist die Platzierung einer entsprechenden Anzahl von Kameras oder ähnlicher Sensoren essentiell. Das Prinzip kann auf die Verwendung von Lichttechnik übertragen werden.

Anzahl der abgetrennten Räume

Werden innerhalb einer Einrichtung beispielsweise Geräte zur Steuerung der Innen-Temperatur verwendet, ist es vorteilhaft, einzelne Zimmer damit auszustatten, da vor allem in Einrichtungen mit

stets geschlossenen Türen mehrere voneinander getrennte Bereiche entstehen, die separat ausgestattet werden müssen. Vor allem in besonderen Einrichtungen wie Kreuzfahrtschiffen oder Hotels ist dieser Faktor signifikant. Auch in diesem Fall kann das Prinzip auf die Verwendung von Lichttechnik übertragen werden, da Türen meist lichtundurchlässig sind.

Frequenz der Zustandsänderungen einzelner Geräte

Offensichtlich ist die Datenrate eines Gerätes ebenfalls ein starker Faktor der Gesamtrate. Die Frequenz, in der Geräte Informationen an das System übermitteln, kann dabei in einem unbegrenzten Spektrum liegen, was meist von der Einsatz-Domäne abhängt.

Größe der Information einer Zustandsänderung

Die Menge an Informationen innerhalb einer zu transferierenden Zustandsänderung ist ebenfalls ein ausschlaggebender Faktor für die Gesamtdatenrate. Dabei kann der Datentyp eine Rolle spielen. Werden Bilder an das System übertragen, resultiert daraus eine höhere Rate als im Fall, in dem ein Sensor bei gleicher Frequenz seine gemessene Temperatur (wenige Bytes) übermittelt.

4.2 Relevanz hoher Performance

Dennoch existieren im Kontext von Smart Home Anwendungen und Funktionen, für welche eine hohe Performance bzw. die Verarbeitung vollständiger Datensätze in korrekter Reihenfolge vorteilhaft oder gar essentiell ist. Einige davon werden im Folgenden aufgezeigt.

Optimieren von Prozessen mittels Algorithmen

Um Prozesse innerhalb des Smart Home Systems zu optimieren, bedarf es der Speicherung von möglichst allen Zustandsänderungen. Denn je mehr Daten für eine nachträgliche Analyse zur Verfügung stehen, desto präziser ist die daraus gewonnene Optimierung.

Historie betrachten

Neben der Analyse mittels Algorithmen kann ebenso die Sicht eines Menschen auf die gesammelten Daten dazu beitragen, Defekte in der Sensorik oder der Automatisierung festzustellen oder eigene Rückschlüsse zu ziehen.

Verarbeitung in Echtzeit

Bei einer hohen Performance in der Aufzeichnung und Verarbeitung kann mittels automatisieren Routinen in Echtzeit auf hochfrequente Informationen reagiert werden. Auf diese Weise werden unerwartete Verhaltensweisen vermieden, da jeder Wert ohne nennenswerte Verzögerung in die Verarbeitung und deren Logik zur Automatisierung einfließen kann.

4.3 Problem der Speichergröße

Mit der Verarbeitung einer hohen Datenrate und dem Speichern dieser erhaltenen Informationen schwingt das Phänomen mit, dass eine bedeutend große Menge von Daten in dem verfügbaren persistenten Speicher abgelegt werden, falls entsprechende Motive für die Analyse oder Verarbeitung im Nachhinein durchgeführt werden sollen. Die Herausforderung bei der Entwicklung einer hoch performanten Smart Home Lösung besteht deshalb neben einer schnellen und zuverlässigen Verarbeitung darin, diesen Speicher sparsam und sinnvoll zu nutzen, um möglichst viele Daten darin abzulegen. Andernfalls entsteht aufgrund dessen, dass diese Ressource erschöpflich ist, ein Verlust an Informationen, wodurch die Funktionalität eingeschränkt werden kann.

4.4 Fazit

Der erforderliche Grad an Performance eines Smart Home Systems ist abhängig von mehreren Eigenschaften der zu verwaltenden Umgebung. Dabei ist nicht nur die Anzahl der inkludierten Geräte entscheidend. Bei der Wahl oder Programmierung eines solchen Systems sollte vorher festgestellt werden, welche der genannten Faktoren auf die Last des Produktes wirken werden. Zudem sollten beim Konzipieren eines Smart Home Systems Mechanismen integriert werden, die die große Menge an Informationen sinnvoll und platzsparend persistent speichern.

5 Related Work

Für die vorgestellten Probleme innerhalb von Smart Home Systemen sowie die Umsetzung eines Benchmarks für solche Software-Produkte existieren bereits einige Lösungen. Im Folgenden wird eine Auswahl davon kategorisch dargelegt.

5.1 Smart Home Benchmark

In [33] wird ein Benchmark für Automatisierungen des Smart Home Dienstes IFFFT durchgeführt. Für den Versuch wird ein eigener Service Provider bei IFTTT registriert, welcher in der Lage ist, mit smarten Geräten und der Plattform selbst zu kommunizieren. Dabei werden Steuerbefehle zum Setzen des Zustandes von IFFFT an die Geräte und gemeldete Zustandsänderungen seitens der Geräte an den Internetdienst weitergeleitet. Die Automatisierung wird mithilfe von Applets so konfiguriert, dass Geräte, welche ihren Zustand beim Auslösen ändern sollen, mit dem eigenen Service Provider verbunden sind. Anschließend werden die Auslöser des Applets getriggert und die Zeit gemessen, die IFFFT benötigt, um die Aktionen bzw. Zustandsänderungen an den Provider zu senden, welcher diese dann an die Geräte schicken wird.

5.2 Eventverarbeitung

Die Verarbeitung von einzelnen Datenpaketen, um Events auszulösen, stellt eine zentrale Aufgabe eines Smart Home Systems dar. Da diese einen relativ hohen Einfluss auf die Performance des Systems als solches haben (im Vergleich zu anderen essentiellen Aufgaben), wird die Abstraktion aller Aufgaben eines Smart Home Systems auf die der Eventverarbeitung als sinnvoll erachtet.

In [13] wird eine Klassifizierung für Systeme zur Verarbeitung von Events eingeführt. Diese sieht die Unterteilung in Active Database Systems, Data Stream Management Systems und Complex Event Processing Systems vor. Dabei sind Active Database Systems eine Erweiterung klassischer Datenbankmanagementsysteme. Jedoch wird bei diesen das reaktive Verhalten von der Anwendungslogik in der Schicht der Datenbank verschoben, wobei drei Kontexte verwendet werden können. In Datenbankerweiterungen verweisen Regeln nur auf den internen State der Datenbank und reagieren automatisch auf Verstöße von festgelegten Auflagen. Innerhalb geschlossenen Datenbankanwendungen unterstützen Regeln die Logik des Programms ohne dabei externe Regelungen einzubeziehen und in offenen Datenbankanwendungen werden sowohl interne als auch externe Semantiken unterstützt. Jedoch sind Active Database Systems für hohe Datenströme ungeeignet. Aus diesen Einschränkungen resultierte die Entwicklung von Data Stream Management Systems. Diese weisen wesentliche Unterschiede in der Beschreibung von eingehenden Daten auf. Zum einen werden keine Beschränkungen von Streams festgelegt. Zum anderen werden keine Aussagen über deren Reihenfolge getroffen. Aufgrund dieser geringen Informationen über die Daten sollen diese nur einmalig verarbeitet werden. Das

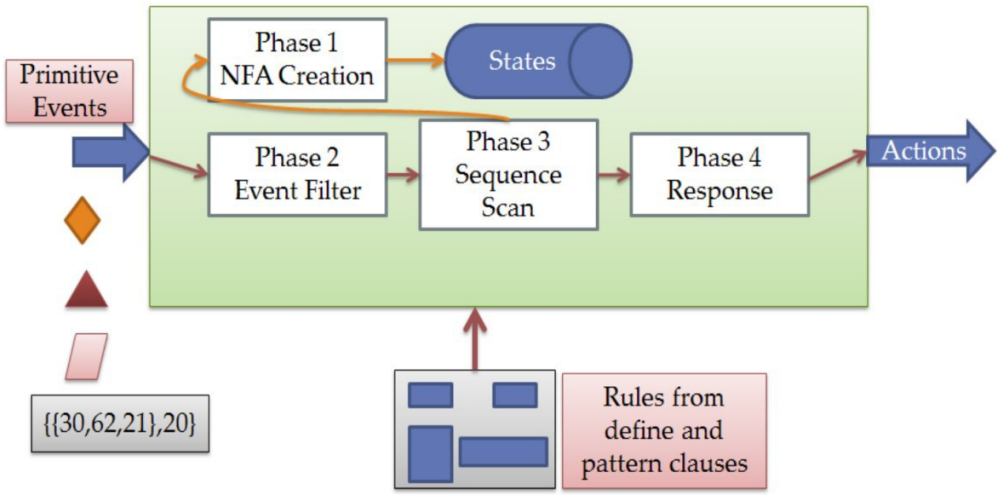

Abbildung 5.1: Funktionsweise verteiltes CEP [41]

heißt, dass Abfragen im Vorhinein festgelegt werden, welche entweder bei Eintreffen von Daten oder periodisch ausgeführt werden. Dadurch müssen die einzelnen Daten nicht selbst abgefragt werden, wenn sie benötigt werden, da die Antwort bereits aufgrund vorher ausgeführter Abfragen berechnet wurde (auch „database-active humanpassive“ (DAHP) genannt). Die dritte Klasse ist die des CEPs, welches das Publish-Subscribe-Verfahren erweitert. Dabei können Senken der CEP-Engine mitteilen, an welcher Art zusammengesetzter Events sie interessiert sind. Die Engine bereitet komplexe Events auf Basis der festgelegten Konfiguration auf, wobei jene Events abhängig vom Auftreten verschiedener einzelner Events abhängt. Der Fokus bei CEP liegt auf der Skallierbarkeit und mehrere Architekturen wie zentralisiert, hierarchisch, azyklisch oder Peer-to-Peer möglich sind.

Omran Saleh von der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau University stellt in der Arbeit [41] einen eigenen Ansatz vor, um komplexe Events zu verarbeiten. Seinerseits werden Konzepte, die auf einer zentralisierten Architektur beruhen, als naiv eingestuft, da in diesen die ankommenden Daten ungefiltert die Hauptinstanz der Verarbeitung passieren. Stattdessen konzentriert sich sein Entwurf auf eine verteilte Verarbeitung mittels unterschiedlicher Knotenpunkte. Dazu werden vier Phasen eingeführt, welche definiert durch unterschliche Aufgaben die Lasten der eigentlichen Instanz der Verarbeitung aufteilen und somit effizienter verarbeitet werden können. Eingehende Daten durchlaufen diese dabei wie in Abbildung 5.1 beschrieben. Demnach werden die Daten zuerst gefiltert, indem überprüft wird, ob vorbestimmte Bedingungen erfüllt sind. Anschließend werden sie einem Sequenz-Scan unterzogen. Bei diesem werden die eingehenden Daten in Abhängigkeit ihrer Zeit des Auftretens mit einem Muster in Form einer Sequenz abgeglichen. Entsprechen sie nicht dem zu erwartenden Ergebnis, werden sie verworfen. Dabei wird vom Nutzer ein Modus aus drei verfügbaren Modi gewählt. Die Auswahl hängt dabei meist von der Anwendungsdomäne ab. Schließt der Sequenz-Scan erfolgreich ab, wird das resultierende Datenpaket an die Senke übergeben. Dort können die komplexen Events weiterverarbeitet werden. Zudem wird ein Nichtdeterministischer endlicher Automat genutzt, um komplexe Sequenzen der Events zu akzeptieren oder nicht. In Abbildung 5.1 ist die grobe Architektur dieser Vorgehensweise zu sehen.
In [53] werden für die Verarbeitung von Events Ontologien verwendet. Ontologien als solche verfolgen den Zweck, Wissen strukturiert zu formalisieren und dienen oft dem Austausch zwischen verschiedenen Anwendungen, um beispielsweise Wissensbestände zusammenzufügen oder in solchen zu suchen. In der erwähnten Arbeit werden die komplexen Ereignisdefinitionen in einzelne Teile zerlegt und in der Ontologie verankert. Daraus kann dann CEP-Programmcode generiert werden, der aus Streams (Verbindungen zu Datenquellen) und Abfragen (tatsächliche Ereigniserkennung) besteht. Um die

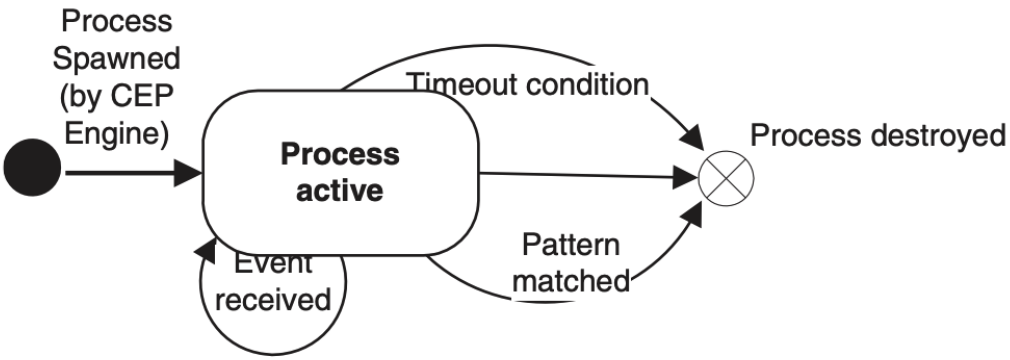

Abbildung 5.2: Zustandsdiagramm gleichzeitiges CEP [29]

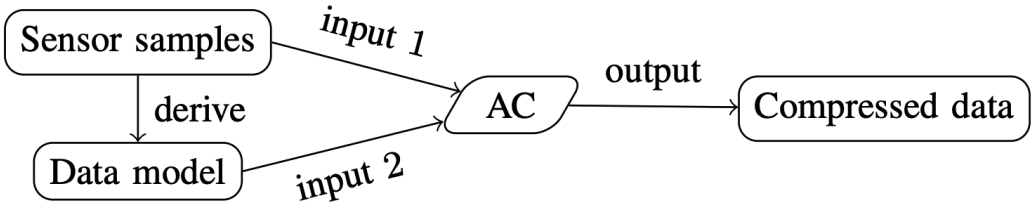

Abbildung 5.3: Komprimierung von Sensor-Daten [51]

Performance des gesamten Systems zu steigern, werden Streams semantisch mithilfe von Reasoning optimiert, d.h. sie können für mehrere Abfragen wiederverwendet werden, falls diese die gleichen Streams abfragen. So können Informationen wiederverwendet und die Datenmenge reduziert werden, wodurch Laufzeitressourcen gespart werden. Das Verfahren wird in der Industrie angewendet, um das Klima der Umgebung von Nahrungspflanzen exakt zu überwachen, sodass Wachstumsbedingungen erforscht werden können.
An der City University von London entstand eine weitere Architektur[29] zur hoch-performanten Verarbeitung von komplexen Events. Der Fokus liegt dabei auf der Sprache Erlang sowie der Open Telecom Plattform, welche für das Arbeiten mit mehreren Prozessen geeignet ist. In dem Ansatz werden die Event-Condition-Action-Regeln als Funktionen implementiert und bei Abruf in einem separaten Prozess abgearbeitet. Nach dessen Beendigung wird darin reservierter Speicher automatisch freigegeben, wodurch kein Garbage Collector notwendig ist. Bedingungen der Regeln können mit den logischen „UND“ und „ODER“ strukturiert werden. Zudem sind auch zeitliche Bedingungen möglich. Diese sind auch in Domäne Smart Home von Relevanz. In Abbildung 5.2 ist das entsprechende Zustandsdiagramm zu sehen.

5.3 Komprimieren von Sensor-Daten

Um eine außerordentliche Menge an Sensor-Daten effizient abzuspeichern, bietet die Arbeit [51] einen Ansatz, solche Informationen verlustfrei zu komprimieren. Die Vorgehensweise basiert auf arithmetischer Kodierung. Dabei werden wie in Abbildung 5.3 Sensor-Proben und ein Datenmodell als Eingabe verwendet. Es wird in zwei Schritten vorgegangen. Zuerst wird eine diskrete Cosinustransformation (DCT) auf die vorhandenen Proben angewandt. Aus den daraus erhaltenen Koeffizienten wird basierend auf deren Energie eine Teilmenge bestimmt. Da das DCT ein verlustbehaftetes Verfahren ist und

bisher noch Fehler inbegriffen sind, werden im zweiten Schritt Ableitungen, welche Approximationsfehler und Sensorrauschen darstellen sollen, modelliert. Dazu wird das weiße Gaußsche Rauschmodell genutzt. Aus den Koeffizienten und den Ableitungen wird anschließend eine Wahrscheinlichkeit bestimmt, dass eine Probe von Sensor-Daten zu einer gewissen Zeit einen bestimmten Wert aus dem Wertebereich annimmt. Aus den Sensor-Informationen und den berechneten Wahrscheinlichkeiten wird dann die arithmetische Kodierung durchgeführt.

6 Konzept

Um den genannten Problemen entgegenzuwirken, soll in diesem Kapitel ein Prototyp eines Smart Home Systems konzipiert werden, dessen Fokus auf der Maximierung der Performance sowie dem damit verbundenen Ziel der Speichereffizienz liegt. Umgesetzt wird der ausgearbeitete Entwurf im nächsten Kapitel.

6.1 Benutzerdefinierte Einheiten

Um dem Anwender die Konfiguration von Geräten, Gruppen und Regeln zu ermöglichen, werden benutzerdefinierte Einheiten eingeführt. Mithilfe von wohlgeformten Strukturen können diese vom Nutzer angegeben werden. Für die Einrichtung dieser legt der Anwender die entsprechenden Dateien in ein Verzeichnis seiner Wahl, dessen Pfad in den Argumenten des Programms untergebracht wird. Die Smart Home Systeme OpenHAB und Eraser ermöglichen ebenfalls das Konfigurieren von Regeln und Geräten mittels Dateien.[39,54]

6.1.1 Geräte

Geräte innerhalb der zu konzipierenden Anwendung können mit physischen Geräten der realen Welt verglichen werden. Sie können mehrere Kanäle enthalten, welche jeweils eigene Zustände annehmen können. Grundsätzlich werden Geräte in zwei Arten untergliedert. Zum einen gibt es „Benutzergeräte“, welche vom Nutzer selbst angelegt werden. Die Konfiguration dieser wird erleichtert, indem dabei pro Gerät ein Topic (für die Kommunikation mittels MQTT) und ein einmaliger Gerätename vergeben wird. Zusätzlich wird jeder Kanal mit einem Namen sowie ggf. einem eigenen Topic beschrieben. Um das gesamte Topic des Gerätes zu erhalten, werden Geräte-Topic und Kanal-Topic getrennt durch ein „/“ konkateniert.

Zudem sollen Systemgeräte einige praktische Grundfunktionen mitbringen. Beispielsweise kann eine benutzerdefinierte Tageszeit Bedingung einer Regel sein. Dazu wird das Systemgerät „Uhr“ verwendet Somit lassen sich Aktionen innerhalb des Smart Home Systems zeitgesteuert automatisieren. Um in OpenHAB von der Tageszeit abhängige Bedingungen in Regeln einzurichten, können darin sogenannte cron-Ausdrücke verwendet werden.[40]

6.1.2 Regeln

Jede Regel besteht aus zwei syntaktisch getrennten Blöcken. Dies sind der Bedingungsblock sowie der Steuerblock.

Bedingungsblock

Ein Bedingungsblock kann aus einer oder mehreren Bedingungen konstruiert sein. Im letzteren Fall sind diese durch Operatoren wie dem logischen „AND“ oder dem „OR“ miteinander verknüpft. Eine Bedingung setzt sich aus vier Attributen zusammen. Dies sind das Gerät, ein Kanal dessen sowie ein Vergleichs-Operator und eine Zeichenkette. Die Bedingung gilt als erfüllt, wenn der Kanal des Gerätes einen Zustand annimmt, der verglichen durch den Operator mit der Zeichenkette auf „wahr“ abbildet. Geschachtelte Bedingungsblöcke gelten hingegen als erfüllt, sobald die einzelnen Bedingungen im Zusammenhang mit den dazwischen befindlichen Verknüpfungen (im Falle von mehreren Bedingungen) nach logischem Auflösen auf „wahr“ abbilden.

Steuerblock

Im zweiten Block der Regel können beliebig viele Aktionen deklariert werden, welche sequentiell abgearbeitet werden, sobald der Bedingungsblock der Regel erfüllt ist. Diese werden daher sequentiell festgelegt. Eine Aktion kann zwei verschiedene Formen annehmen. Zum einen kann der Zustand eines beliebigen angemeldeten Gerätes geändert werden. Dieser kann dabei neu definiert oder von Zuständen anderer Kanäle (auch anderer Geräte) abhängig gemacht werden. Im letzteren Fall kann dabei der aktuelle Wert oder ein historischer Wert referenziert werden. Zum anderen kann eine Aktion aus dem Auslösen einer internen Funktion bestehen. Interne Aktionen sind das Schreiben in die Log-Datei oder das Veröffentlichen einer unabhängigen MQTT-Nachricht.

6.1.3 Datengruppen

Mithilfe von Datengruppen können Profile erstellt werden, welche die Präzision der zu verarbeitenden und zu speichernden Zustandsänderungen bestimmen. Dabei werden pro Gruppe zwei Dezimalzahlen angegeben. Dies sind die Lese- und die Schreibfrequenz. Jeder Kanal kann höchstens einer Datengruppe angehören. Deren Referenz wird in der Struktur des Kanals untergebracht.

Lesefrequenz

Die Lesefrequenz gibt an, in welcher Frequenz erhaltene Zustandsänderungen verarbeitet werden. Ziel ist es, unnötig hochfrequente Informationen noch vor deren Verarbeitung zu verwerfen, so dass Rechenleistung gespart wird.

Schreibfrequenz

Um Speicherengpässe einzuschränken, wirkt die Schreibfrequenz als Stellschraube bei der Zusammenfassung von Zustandsänderungen in der Datenbank. Dabei werden mehrere Werte innerhalb einer Periode dieser Frequenz zu einem Wert zusammengefasst. Das genaue Vorgehen wird in diesem Kapitel unter „Benutzerdefiniertes Zusammenfassen von Daten“ genauer erläutert.

6.2 Zielorientierte Methodiken

Zunächst werden Methodiken beschrieben, die bei der Konzipierung angewandt werden, um die gegebenen Probleme und Herausforderungen zu bewältigen.

6.2.1 Maximieren der Performance

Die Performance des zu erstellenden Programms soll maximal sein. Mit folgenden Elementen des Konzepts wird auf dieses Ziel zugearbeitet.

Laufzeit und Sprache

Ein ausschlaggebender Faktor, welcher die Performance beeinflusst, ist die verwendete Programmiersprache und deren dedizierte Laufzeitumgebung. Um sie zu steigern, eignet sich das Verwenden einer Sprache, deren Code sich in prozessornahen Maschinencode kompilieren lässt. Zum Ausführen solcher Programme wird keine installierte virtuelle Maschine gebraucht, denn das Betriebssystem stellt dabei die Laufzeitumgebung dar. Folglich werden genutzte Ressourcen wie CPU und RAM nicht von virtuellen Maschinen belastet und können für den eigentlichen Programmablauf verwendet werden. Zu solchen Programmiersprachen zählen unter anderem C, C++ und Rust.

Datenbank

Hinsichtlich des persistenten Speicherns von Zustandsänderungen sowie anderen essentiellen Informationen ergeben sich zwei infrage kommende Ansätze. Zum einen bieten SQL-Datenbanken die Möglichkeit, zu speichernde Inhalte geordnet nach Relation strukturiert abzulegen. Auf der anderen Seite sind sie nach den Ergebnissen von [4] langsamer in der Bearbeitung von gleichzeitigen Abfragen. Andererseits lassen sich Daten mithilfe von NoSQL-Datenbanken weniger relational, jedoch zügiger abspeichern.[58] Aufgrund dieses Unterschieds fällt die Entscheidung auf das Verwenden einer NoSQL-Datenbank. Resultierend daraus schwindet die Garantie auf einen vollständig konsistenten Datensatz. Da die Relation der Daten in diesem Fall keine große Relevanz hat, weil nur es nur zwei voneinander unabhängige Sammlungen geben soll, folgt daraus, dass die gewünschte Funktionstüchtigkeit der Software nicht beeinflusst wird.

Datenstruktur

Für schnellen Zugriff auf relevante Objekte und Strukturen, werden Geräte und Regeln auf besondere Weisen verwaltet. Jedes Gerät existiert genau einmalig und wird während der Laufzeit mit relevanten Daten aktualisiert. Dementsprechend verfügt jedes Geräte-Objekt über die Instanz seines Treibers und jeder Kanal über die Instanz seiner Datengruppe sowie einer Liste von Regeln, in deren Bedingung er von Relevanz ist. Sobald sich der Zustandswert eines Kanals ändert, wird dieser Wert und dessen Zeit des Auftretens in der Datenstruktur überschrieben.

Regeln

Nach dem Parsen der Regeln liegen deren Bedingungen als Baumstruktur vor. Beim Auswerten dieser werden aus der Datenbank geladene Ergebnisse in einer dafür initialisierten Variablen gesichert und können somit von allen prüfenden Funktionen mehrmals verwendet werden, ohne die Abfrage mehrmals auszuführen. Jede Aktion einer Regel wird als Liste gespeichert, in welcher ein Zeiger auf die auszuführende Funktion die erste Position einnimmt und dahinter die Argumente folgen. Zudem werden Regeln, welche zeitabhängige Bedingungen enthalten, dahingehend überprüft, ob die Bedingungen mit dieser Eigenschaft alleinige Auslöser der Regel sein könnten. Dabei wird eine Rekursion über die Baumstruktur der Bedingungen ausgeführt, wobei „AND“-Knoten die kleinste Liste an zeitlich abhängigen Bedingungen an den Elternknoten überreichen und „OR“-Knoten eine konkatenierte Liste. Enthält die Liste der Wurzel demnach Elemente, werden deren Zeiten extrahiert und für das Setzen eines Timers genutzt, nach dessen Ablauf die Regel ausgelöst wird. So wird vermieden, dass nach jeder Änderung der Tageszeit überprüft werden muss, ob Regeln ausgeführt werden müssen.

Weiterverarbeitung der Zustandsänderung in channel-spezifischer Lese-Frequenz

Durch den Einsatz der Datengruppen werden Zustandsänderungen, deren Zeit des Auftretens kleiner ist als das Intervall der Lese-Frequenz, addiert mit der Zeit der letzten Zustandsänderung, verworfen.

Dadurch bleiben das Überprüfen, ob Regeln ausgeführt werden müssen sowie das Speichern in der persistenten Schicht aus, so dass Rechenressourcen gespart werden

Überprüfen auf tatsächliche Änderung

Bei Erhalten einer Zustandsänderung wird überprüft, ob das Gerät sowie dessen entsprechender Kanal bereits diesen Zustand angenommen hat. Bei positivem Ergebnis kann Rechenleistung gespart werden, da nachfolgende Prozesse wie die Eventverarbeitung oder das Abspeichern der Information in der Datenbank nicht stattfinden müssen.

6.2.2 Maximieren der Speichereffizienz

Das zu konzipierende System soll in der Lage sein, hochfrequente Zustandsänderungen ohne Probleme entgegenzunehmen. Diese Informationen können genutzt werden, um in Zukunft Optimierungsverfahren durchzuführen. Folglich ist es vorteilhaft, jene Daten zu speichern. Aufgrund der Tatsache, dass jeder persistente Speicher endlich ist, muss die große Menge solcher Daten effizient und speicherschonend abgelegt werden. Im Folgenden werden diesbezüglich einige Techniken erläutert.

Benutzerdefiniertes Zusammenfassen von Daten

Durch die hohe Frequenz eingehender Informationen müssen große Datenmengen verwaltet und sinnvoll abgespeichert werden. Dabei soll der Informationsgehalt der Daten ausreichend erhalten bleiben. Um einen Kompromiss zwischen Präzision und Speichereinsparung zu schließen, werden empfangene Zustandsänderungen in einem separaten Workflow passend und kontextbezogen zusammengefasst. Der Grad an Komprimierung hängt von der Konfiguration der jeweiligen Datengruppe ab, welche für jeden Channel eines Gerätes festgelegt werden kann, aber nicht zwingend muss. Für das benutzerdefinierte Zusammenfassen der Daten eines Geräte-Kanals werden zunächst Abfragen zum Extrahieren einer festgelegten Anzahl von dessen ältesten Zuständen in der Datenbankschicht ausgeführt. Anschließend wird die Schreib-Frequenz der Datengruppe verwendet, um das Intervall dieser zu berechnen. Danach wird das erste Intervall seit dem Start des Programms berechnet, um so das älteste Intervall, in denen sich abgefragte Daten zeitlich einordnen lassen, zu erhalten. Alle Zustände, die Teil dieses Datensatzes sind, werden dann mithilfe des arithmetischen Mittels zusammengefasst. Nach dieser Aggregation, wird der zusammengefasste Eintrag mit der Anzahl der ursprünglich verwendeten Einträge sowie der Schreib-Frequenz in einer zweiten Sammlung („Historie“) abgelegt, wobei die benutzte Datenmenge aus der ersten Sammlung („Live“) entfernt wird. Dieser Ablauf wird anschließend für die darauffolgenden Intervalle durchgeführt bis die initial geladene Menge an Zuständen vollständig abgearbeitet werden konnte oder das Intervallende in der Zukunft liegt. Dieser Vorgang wird in einem separaten Thread abgehandelt und in regelmäßigen Zeitabständen nach dem letzten Durchlauf gestartet, so dass nie zwei dieser Threads gleichzeitig arbeiten.

Datenbank

Auch die Speichereinsparung wird durch die Anbindung der NoSQL-basierten Datenbank erhöht. Denn jene Strukturen besitzen neben der hohen Performance die Eigenschaft, mit steigender Datenmenge weniger Speicher als SQL-Datenbanken zu verbrauchen.[9]

Überprüfen auf tatsächliche Änderung

Empfangene Änderungen werden vor dem Abspeichern mit dem aktuellen Zustand verglichen und bei Gleichheit verworfen. Dies soll verhindern, dass irrelevante Zustandsänderungen unnötigen Speicherplatz in der Datenbank einnehmen.

6.3 Funktionale Anforderungen

Einlesen von Geräten und Regeln

Vom Benutzer angelegte Geräte und Regeln (siehe Komponenten) werden vom Programm eingelesen und können danach verwendet werden.

Empfangen von Zustandsänderungen mittels MQTT

Während der Laufzeit wartet das Programm auf eingehende Zustandsänderungen. Dazu wird ein MQTT-Client so konfiguriert, dass jede Änderung eines angemeldeten Gerätes entgegengenommen wird. Anschließend wird diese persistent abgelegt.

Persistentes Verwalten empfangener Zustandsänderungen

Für die nachträgliche Auswertung und Integration von historischen Zuständen in die Verarbeitung von Regeln, müssen diese persistent festgehalten werden und für Abfragen bereitstehen. Dazu werden zwei Tabellen benötigt.

Regeln abarbeiten

Der Benutzer soll dabei unterstützt werden, Reaktionen auf Zustandsänderungen beliebiger Geräte automatisch auslösen zu können. Dies dient dem Zweck, das Gebäude oder Gelände autonom zu steuern. Aus diesem Grund werden Regeln ein essentieller Bestandteil der Software sein. Innerhalb dieser sollen benutzerdefinierte Abläufe an Bedingungen geknüpft sein (vgl. OpenHAB[39]) Jene Abläufe beinhalten auch das Übertragen eines Zustandes vom Zeitpunkt der Vergangenheit (relativ zur Zeit der Aktivierung der Regel) von einem Geräte-Kanal auf einen anderen.

Ändern von Zustandsänderungen mittels MQTT

Soll durch das System eine Änderung eines Gerätes ausgelöst werden, erstellt und versendet das Programm eine Nachricht auf MQTT-Basis, um diese Information an das entsprechende Gerät weiter zu leiten.

Empfangen von manuellen Befehlen (API)

Ein Endpunkt im System soll es ermöglichen, direkt Befehle vom Benutzer zu erhalten und ihm Informationen bereitzustellen. Dazu zählen das manuelle Ansteuern von Geräten und Auslesen eines aktuellen Gerätzustandes oder das Auslösen einer Regel. Weiterhin kann er die vollständige Liste von Regeln oder Geräten abfragen. Umgesetzt wird die Schnittstelle durch eine REST-API.

6.4 Einschränkungen

Beim Entwickeln der Software wird auf Elemente verzichtet, welche zwar nicht relevant für die gewünschte Lösung sind, jedoch Bestand in der Realität haben, wenn ein marktfähiges Smart Home Produkt konzipiert wird. Das Weglassen dieser hat keinen fälschenden Einfluss auf die beabsichtigte Steigerung der Performance.

Fehlende Items

Auf die Implementation von Items, (vgl. OpenHAB[28]) wird verzichtet, da diese für das Modell der Automatisierung nicht benötigt werden.

Keine hohe Fehlertoleranz

In der Konzeption und Umsetzung wird die Fehlerbehandlung auf ein Mindestmaß beschränkt. Im Falle eines unerwarteten Ereignisses oder Wertes kann dieses nur verworfen werden. Sind die benutzerdefinierten Einheiten nicht syntaktisch wohlgeformt, kann es zu Fehlern im Programmablauf kommen.

Sicherheit

Die Software kann netzwerkweit von jedem Benutzer angesprochen werden und es werden keine Maßnahmen zur Autorisierung getroffen wie dies z. B. bei dem SHS „Google Home“ der Fall ist.[60]

Medien

Videos, Bilder und Audiomedien können nicht als solche verarbeitet werden, da es nicht für einen Proof-Of-Concept notwendig ist, Streams o.ä. einzubinden oder zu verarbeiten. Umsetzbar wäre die Funktionalität durch Kodierung der Medien in Zeichenketten, um sie dann wie andere Zustände zu verarbeiten. OpenHAB hingegen unterstützt das Empfangen solcher Elemente durch URLs.[44]

Kontinuierliches Einlesen benutzerdefinierter Einheiten

Vom Benutzer erstellte Regeln oder Geräte müssen vor Programmstart ordnungsgemäß konfiguriert sein. Ein Einlesen oder Aktualisieren, wie es OpenHAB praktiziert[12], wird während der Laufzeit nicht unterstützt. Da dies nur in einem weiteren Thread ablaufen würde, hätte es keinen negativen Einfluss auf die Performance der Event-Verarbeitung.

6.5 Vergleich zu Related Work

Das Ablegen des aktuellen Zustands in der Struktur eines Kanals kann mit der Funktionsweise von Data Stream Management System, welche in [13] vorgestellt werden, verglichen werden. Der Zustands wird bei Eintreffen einer Änderung in dem entsprechenden Kanal aktualisiert um so für Abfragen, nach dessen Zustands sofort bereit zu stehen. Auf der anderen Seite müssen auch historische Daten zur Verfügung stehen. Dafür bietet ein klassisches Datenbankmanagementsystem genutzt. Weiterhin durchlaufen empfangene Zustandsänderungen einen Workflow, bei welchem bestimmte Kriterien erfüllt sein, müssen um tatsächlich als Event verarbeitet zu werden. Beispielsweise wird überprüft, ob die Frequenz der Lese-Frequenz aus der Datengruppe eingehalten wird, falls diese konfiguriert ist, und, ob ein Duplikat des aktuellen Zustands des relevanten Kanals empfangen wurde. Dieses Vorgehen, was einem Filter gleicht, lässt sich in CEP einordnen, da die Verarbeitung der eingehenden Daten von bereits erhaltenen abhängig gemacht wird. Der Unterschied zu [41] besteht darin, dass dort eine dezentrale Architektur verwendet wird, um die Last auf Ressourcen besser zu verteilen.

7 Umsetzung

Nun soll die konzipierte Anwendung in Form eines funktionsfähigen Prototyps beschrieben und implementiert werden. Zu diesem Zweck werden in diesem Kapitel die zu verwendenden Technologien sowie der Programmablauf und dessen Bestandteile beschrieben. Auch die Umsetzung individueller Anpassungsmöglichkeiten seitens der Nutzer und das zugrunde liegende Modell der Datenbank werden dargelegt. Das zu implementierende Produkt trägt den Namen „Shunoled“

7.1 Benutzerdefinierte Einheiten

Für die Realisierung der benutzerdefinierten Einheiten soll die Notation Javascript Object Notation (JSON) genutzt werden. Jene bietet den Vorteil, weniger Zeit bei der maschinellen Verarbeitung zu beanspruchen als andere Auszeichnungssprachen der Datenserialisierung wie z.B. XML (Extensible Markup Language).[22,26] Zudem ist die Notation für das menschliche Auge wesentlich besser lesbar.[26] Geräte werden demnach mit einer Liste von Channels und den jeweiligen Attributen des Geräts versehen. Jeder Channel gibt maximal eine Referenz einer Datengruppe an, der er angehören soll. Datengruppen enthalten neben einer ID eine Lese- sowie eine Schreibfrequenz. Regeln sind aus einem Bedingungs- und einem Steuerblock aufgebaut, wobei der Steuerblock durch eine Liste eine Reihe an Aktionen definiert. Jede Aktion ist dabei ein eigenes Objekt in der JSON Struktur.

7.2 Module

Das Projekt wird in mehrere Module unterteilt, um die Erweiterbarkeit sowie Übersichtlichkeit zu wahren. Im Folgenden werden diese aufgezählt und kurz deren Funktion erklärt.

Persistenz

Dieses Modul stellt eine Schnittstelle zur Abfrage von sowohl aktuellen als auch historischen Zuständen von Geräten dar. Diese werden aus der angebundenen Datenbank bezogen. Außerdem erhält das Modul ebenfalls eingehende Zustandsänderungen, um sie neben den bereits gespeicherten persistent abzulegen. Weiterhin beinhaltet das Modul die Funktionalitäten des parallelen Threads, welcher für das Zusammenfassen von persistent gespeicherten Daten verantwortlich ist.

Parsen

In dem Modul „Parsen“ werden die vom Nutzer bereitgestellten Dateien für die gewünschte Konfiguration von Geräten und Gruppen eingelesen und in die jeweilige Datenstruktur konvertiert, um während des Betriebs schnellen Zugriff auf sie zu erhalten.

Regeln

Das Einlesen und die Verarbeitung von Gruppen erfolgen in dem Modul „Regeln“. Die darin enthaltene Engine für das konzipierte CEP erhält Zustandsänderungen und überprüft dann, ob die eingerichteten Regeln relevant sind, um danach ggf. deren Aktionen aus dem Steuerblock auszuführen.

REST

Das Modul „Rest“ beinhaltet alle notwendigen Funktionalitäten zum Bereitstellen einer Schnittelle auf HTTP-Basis, sodass eingehende Anfragen beantwortet werden können.

Engine

Die Engine dient dem allgemeinen Programmablauf und ist hauptverantwortlich für die Routine beim Start der Software. Sie enthält zentrale Funktionen, welche als Schnittstelle für andere Module dienen wie z.B. das Setzen oder Abfragen von Gerätezuständen.

Treiber

Da die Software durch weitere Gerätetypen erweitert werden können soll, bietet das Modul „Treiber“ Funktionen, die in einer Instanz eines eigentlichen Treibers implementiert werden müssen, um eine korrekte Einbindung zu garantieren. Jene Funktionen ermöglichen die Kommunikation zwischen den entsprechenden Geräten des Treibers und der Logik.

MQTT-Client

Da nach der Implementation der Software bereits Geräte verwendet werden sollen, stellt ein MQTT-Client beispielhaft einen Treiber dar. So kann ein Benchmark mit simulierten Geräten mittels MQTT-Protokolls durchgeführt werden.

7.3 Modell der Datenbank

Lediglich die übermittelten Zustandsänderungen der Geräte sollen in der Datenbank gespeichert werden. Dafür werden zwei unterschiedliche Tabellen eingeführt. Die Tabelle „Live“ enthält die Spalten „ID“, „Gerät“, „Kanal“, „Zustand“ und „Zeit“. Jede Zustandsänderung wird dort zunächst unmittelbar nach Erhalten abgelegt. Die zweite Tabelle „Historie“ dient dem Zweck, trotz einer großen Masse an übermittelten Informationen Speicher einzusparen. Dazu werden in der „Live“ persistent gehaltene Zustände während der Laufzeit abhängig von dem Faktor der Gruppe des Gerätekanals zusammengefasst und nach „Historie“ verschoben. Der Faktor wird dabei ebenfalls gespeichert. Folglich besteht diese Tabelle aus den gleichen Spalten und wird um das Attribut „Faktor“ erweitert.

7.4 Technologien

Sprache

Für die Implementation der Software wird Python Version 3.6 verwendet. Dies hat den Grund, dass mit dieser Sprache zügig ein Proof-Of-Concept Produkt bereitsteht, welches alle notwendigen Anforderungen befriedigt. Der Code kann zudem übersichtlicher strukturiert werden als bei der Verwendung von Programmiersprachen wie C, C++ oder Rust.[21] Auch Entwickler profitieren von diesem Vorteil bei der Entwicklung von weiteren Treibern. Bei der Entwicklung der Anwendung mithilfe der zuletzt genannten Sprachen bedarf es eines höheren Zeitaufwands. Auf der anderen Seite würden jene Sprachen eine Verbesserung der Performance bewirken.[6] Um dennoch eine solche Erhöhung zu

erreichen, wird der resultierende Code kompiliert.[38] Jedes der zu implementierenden Komponenten lässt sich auch in einer performanteren Sprache wie C oder C++ umsetzen. Da jedoch Effektivität der konzipierten Methoden für die Optimierung der Performance im Vordergrund stehen sollen, wird aus Zeitgründen Python verwendet.

Datenbank

Die Realisierung der persistenten Schicht soll MongoDB eingesetzt werden. Die Software bietet eine NoSQL-Datenbank sowie eine entsprechende Schnittstelle für Python, um zu dessen Laufzeit Transaktionen und Abfragen auszuführen. Zudem verfügt es über performante Kapazitäten.[30]

JSON-Parser

Das Parsen der Dateien mit JSON-Inhalt übernimmt der in Python integrierte JSON-Parser. Die Performance beim Parsen von JSON-Dokumenten ist bei der Umsetzung nicht von Relevanz, da dies lediglich beim Programmstart geschieht und dort in die jeweiligen Datenstrukturen konvertiert wird. Nach diesem Vorgang wird nicht mehr mit den JSON-Dateien gearbeitet.

REST-Framework

Als REST-Schnittstelle wird das Framework „Sanic“ einbezogen. Dies lieferte in Benchmarks[16,55,10] eine für diesen Zweck ausreichende Performance.

MQTT-Client

Die Datenübertragung mittels MQTT wird realisiert durch die Bibliothek „paho-mqtt“. In Benchmarks[8] erwies sich diese für das Vorhaben als ausreichend performant.

7.5 Programmverhalten

7.5.1 Programmstart

Für das Ausführen des Programms sind mehrere Parameter notwendig. Dazu zählen die Adresse der Datenbank sowie der Pfad zu den benutzerdefinierten Einheiten. Nach dem Start werden die folgenden Aktionen ausgeführt:

1. Verarbeiten der benutzerdefinierten Einheiten:
   Um die Konfiguration von Geräten, Regeln und Gruppen in den Programmablauf einfließen zu lassen, werden zunächst verschiedene Parser verwendet, um diese aus dem Dateisystem einzulesen. Die Einheiten werden danach für die Weiterverarbeitung in geeigneten Datenstrukturen abgelegt und stehen für den Gebrauch zur Verfügung.

2. Initialisieren der Datenbankschnittstelle:
   Nun wird die Adresse der bereits eingerichteten Datenbank genutzt, um eine Verbindung zu jener aufzubauen. Anschließend wird die Konfiguration der Gruppen genutzt, um den parallelen Thread zur Datenzusammenfassung einzurichten. Dieser wird danach gestartet.

3. Initialisieren der Treiber (incl. MQTT):
   Als nächstes werden alle von der Software unterstützen Treiber, welche zur Kommunikation unterschiedlicher Geräte dienen, eingerichtet. Jeder Treiber erhält dazu diejenigen Geräte, welche ihn in der Konfiguration der Geräte referenzieren. Nachdem dies erfolgreich abgeschlossen ist, wird jeder Treiber gestartet. Als Beispiel wird in der Programmierung der Software eine MQTT-Schnittstelle als solcher implementiert. Nachdem der Treiber alle Geräte, welche mittels MQTT mit dem System kommunizieren sollen, erhalten hat, wird über diese iteriert und jeweils

das entsprechende Topic gemäß dem Protokoll abonniert, um Zustandsänderungen signalisiert zu erhalten.

4. Starten des parallelen Threads für das Zusammenfassen gesammelter Daten:
   Als letztes startet ein weiterer Thread, um im vorgesehenen Intervall Informationen zusammenzufassen.

7.5.2 Verhalten bei eingehenden Zustandsänderungen

Nachdem eine Zustandsänderung innerhalb eingegangen und mit dem Zeitstempel des Eintreffens sowie der Referenz für Gerät und Channel an die innere Logik der Anwendung übermittelt worden ist, beginnt das Abarbeiten eines Workflows. Dieser sieht zunächst eine Überprüfung vor, ob dieser Channel des Gerätes bereits diesen Zustand angenommen hat. Ist dem so, wird der Workflow beendet. Andernfalls wird als nächstes kontrolliert, ob der Zeitpunkt der zuletzt gemessenen Zustandsänderung weit genug entfernt ist, um die Lesefrequenz der Datengruppe (falls der Channel einer solchen angehört) einzuhalten. Falls diese Bedingung ebenfalls erfüllt ist, wird die Änderung mit dem Zeitstempel in der Datenbank festgehalten und überprüft, ob Regeln angewendet werden müssen.

7.5.3 Verhalten bei eingehenden REST-API Anfragen

Wird eine Anfrage per REST-API gestellt, werden die entsprechenden Module genutzt, um entweder erhaltene Steuerbefehle auszuführen oder Strukturen für die Übermittlung von Informationen zu erstellen, welche als Antwort an den Client gesendet werden.

8 Evaluation

Dieses Kapitel dient der Erforschung messbarer Metriken, welche die Qualität der umgesetzten Software in Form von Auswertungen zeigen sollen.

8.1 Optimierung der Performance

Als erstes soll evaluiert werden, inwieweit die konzipierten und realisierten Mechanismen die Performance der Software positiv beeinflussen, indem die verarbeitende Datenrate gesteigert wird. Ein Vergleich mit den Smart Home Systemen OpenHAB und Eraser soll dazu beitragen, diese Leistung einzuordnen.

8.1.1 Allgemeiner Benchmark

Um die allgemeine Performance des implementierten Smart Home Systems Shunoled zu evaluieren und mit jener der beiden vorgestellten Systeme OpenHAB und Eraser zu vergleichen, wird ein Benchmark auf die gleiche Weise durchgeführt wie bei jenen Systemen. Während des Ablaufs ist die Mechanik zum Zusammenfassen der Daten in der Datenbank ausgeschaltet. Es sollen die bereits verwendeten Konfigurationen zur Messung dienen. Für jede dieser werden erneut fünf Messungen durchgeführt, deren Ergebnisse im folgenden Box-Plot abgebildet sind.

Wie in den Diagrammen aus den Abbildungen 8.1 und 8.2 zu erkennen, kristallisieren sich signifikante Unterschiede in der Performance erst in höheren Datenraten heraus. Auffallend ist, dass Shunoled bei Konfigurationen, in denen die Geräteanzahl wesentlich höher ist als die Änderungsfrequenz pro Gerät, eine höhere Verarbeitungsfrequenz als Eraser verzeichnet und dabei eine geringere Verarbeitungszeit beansprucht. Bei anderen hingegen erreicht Eraser die bessere Leistung.

8.1.2 Effektivität Check auf tatsächliche Änderung

Nun soll der Einfluss des besonderen Umgangs mit Duplikaten von sequentiellen Zustandsänderungen untersucht werden. Dazu wird das geschriebene Benchmark-Werkzeug durch eine Funktion erweitert. Diese provoziert genau dieses Verhalten, indem Zustandsänderungen mehrfach an das zu messende System gesendet werden. Die Anzahl der Duplikate kann vor dem Start des Werkezugs festgelegt werden. Für das aktuelle Experiment wird diese Zahl auf drei festgelegt. Somit wird jeder Zustand insgesamt vier Mal versandt. Aufgrund des Versagens seitens OpenHAB bei der vollständigen Verarbeitung höherer Datenraten wird das System aus diesem Versuch herausgenommen, da das Ergebnis nicht aussagekräftig wäre. Bei der Beschriftung des Box-Plots ist zu beachten, dass die Änderungsfrequenzen der Achse für die Konfigurationen nicht die Duplikate der Zustandsänderungen einschließen. Lediglich die „echten“ Zustandsänderungen sind inbegriffen. Folgende Messergebnisse resultieren aus der Untersuchung von Eraser und Shunoled.

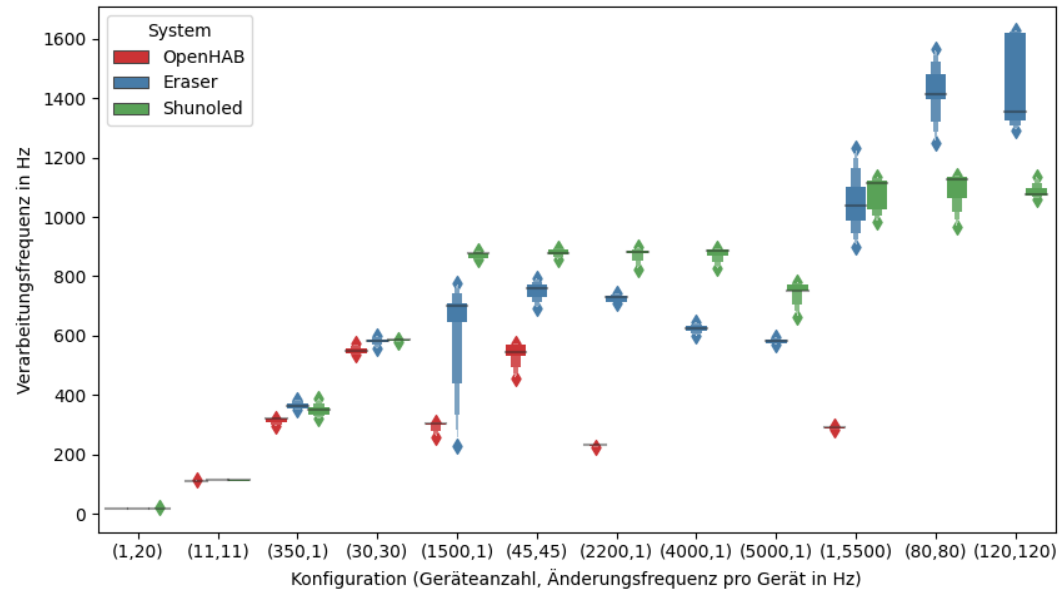

Abbildung 8.1: Verarbeitungsfrequenzen OpenHAB, Eraser und Shunoled

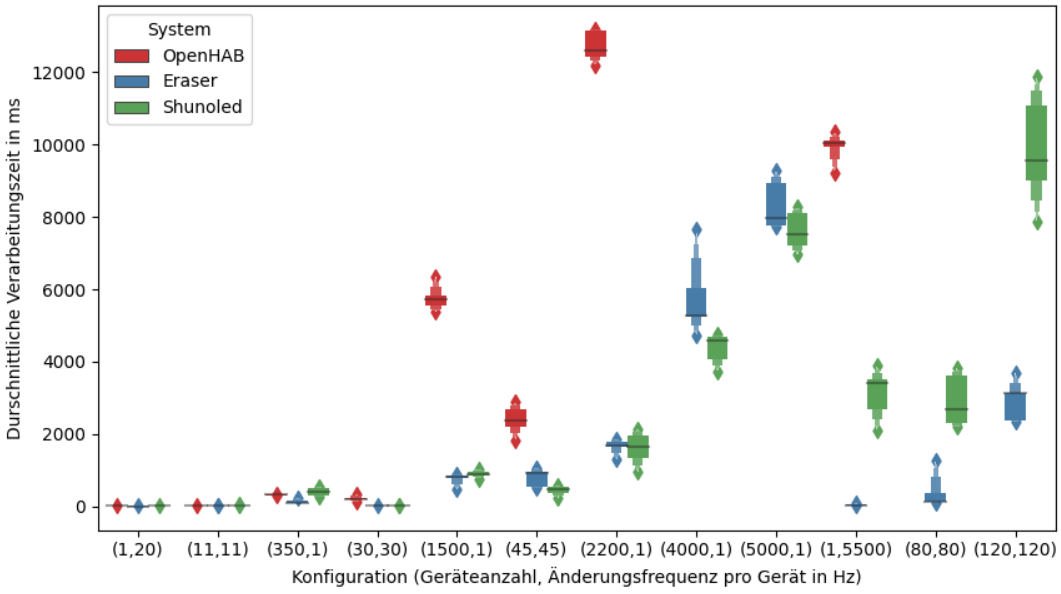

Abbildung 8.2: Verarbeitungszeiten OpenHAB, Eraser und Shunoled

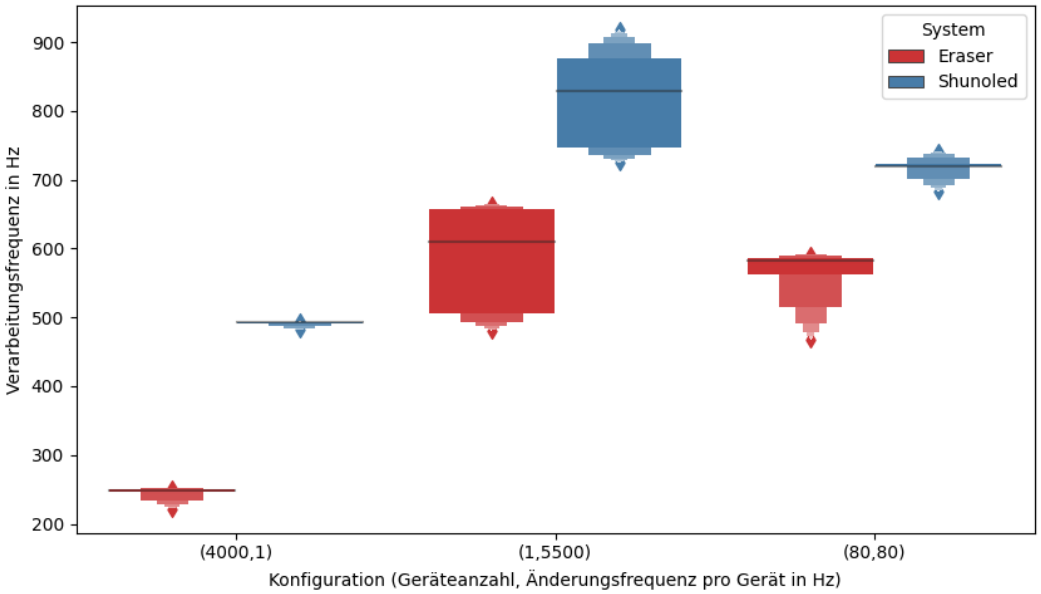

Abbildung 8.3: Verarbeitungsfrequenzen der drei Systeme mit Duplikaten

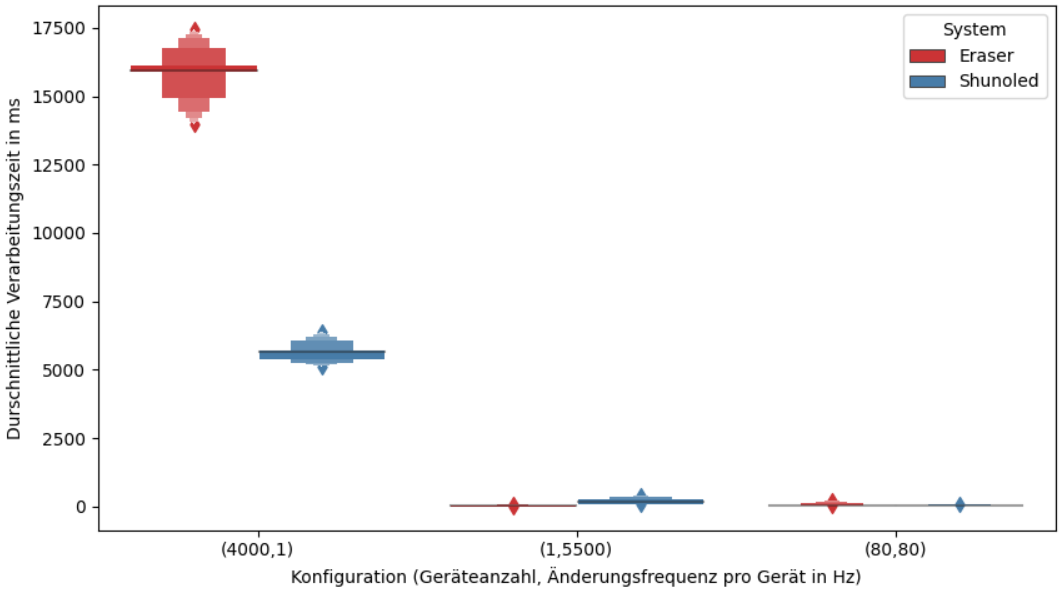

Abbildung 8.4: Verabeitungszeiten der drei Systeme mit Duplikaten

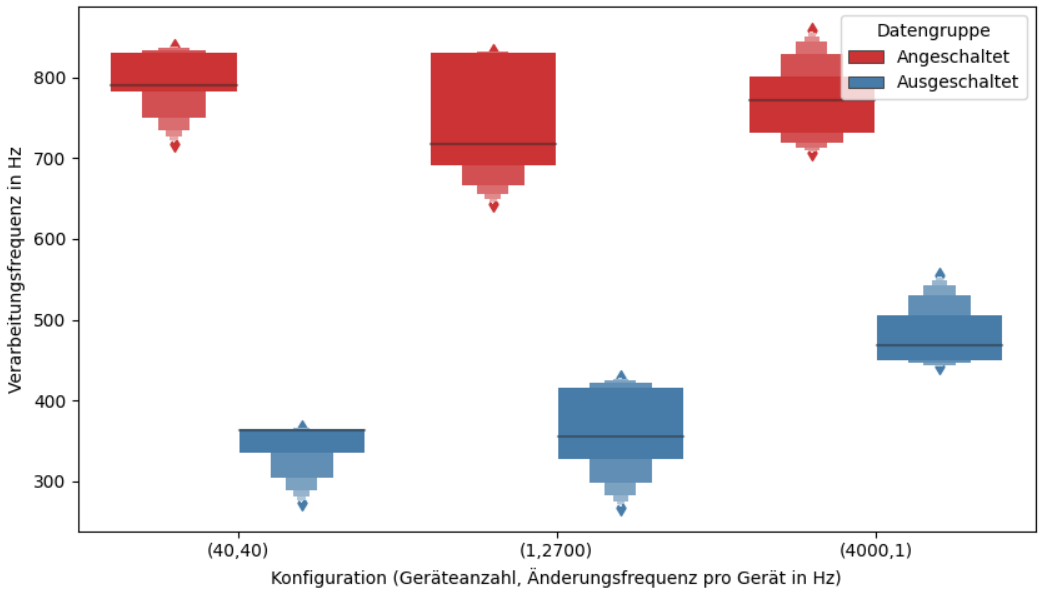

Abbildung 8.5: Verarbeitungsfrequenz mit und ohne Lese-Faktor in einer Datengruppe

Aufgrund des Verwerfens von bereits gesetzten Zuständen und der damit eingesparten Rechenleistung, welche andernfalls für die Eventverarbeitung sowie das Abspeichern in der Datenbank benötigt worden wäre, erzielt Shunoled in allen Konfigurationen sowohl hinsichtlich der durchschnittlichen Verarbeitungszeit als auch der Verarbeitungsfrequenz die besseren Ergebnisse (siehe Abbildung 8.3 und 8.4) Wenn Sensoren bzw. Geräte den Zustand eines ihrer Kanäle dem System periodisch mitteilen, ohne dass sich dieser dabei ändert, kann das System diesen folglich effizienter hinsichtlich der Rechenleistung verarbeiten.

8.1.3 Effektivität der Datengruppen

Um die Steigerung der Performance mithilfe der Datengruppen und deren Lesefrequenz zu zeigen, werden zwei Benchmark Durchgänge gleichzeitig durchgeführt. Dabei soll die Benchmark-Konfiguration des ersten variabel sein und dessen Verarbeitungsfrequenz sowie die durchschnittliche Verarbeitungszeit gemessen werden. Der zweite Durchgang soll eine konstante Belastung des Systems simulieren, indem dieser eine Konfiguration von 40 Geräten und einer Zustandsänderungsfrequenz von 100 Hz pro Gerät annimmt. Dessen gemessene Ergebnisse sind nicht von Relevanz. Die Effektivität der Datengruppen wird mithilfe eines Vergleichs gezeigt, bei welchem die Geräte der zweiten Konfiguration entweder in einer Datengruppe mit einer Lesefrequenz von 1 Hz sind oder sie keiner Datengruppe angehören. Jeder Benchmark dauert fünf Sekunden an.

Resultierend aus den Messwerten in den Abbildungen 8.5 und 8.6 ist eine Verbesserung der Performance erkennbar. Sowohl die Zeit der Verarbeitung als auch deren Frequenz werden im ersten Benchmark gesteigert, wenn die Datengruppe im zweiten Benchmark aktiviert ist. Dadurch wird dem Nutzer die Möglichkeit geboten, mithilfe der Konfiguration von Lesefrequenzen die Datenraten der einzelnen Geräte-Kanäle kontextbezogen und dynamisch anzupassen, wodurch die Gesamtperformance des Systems erhöht wird.

8.2 Optimierung der Speichereffizienz

Die realisierten Methodiken zur Verbesserung der Speichereffizienz werden nun evaluiert.

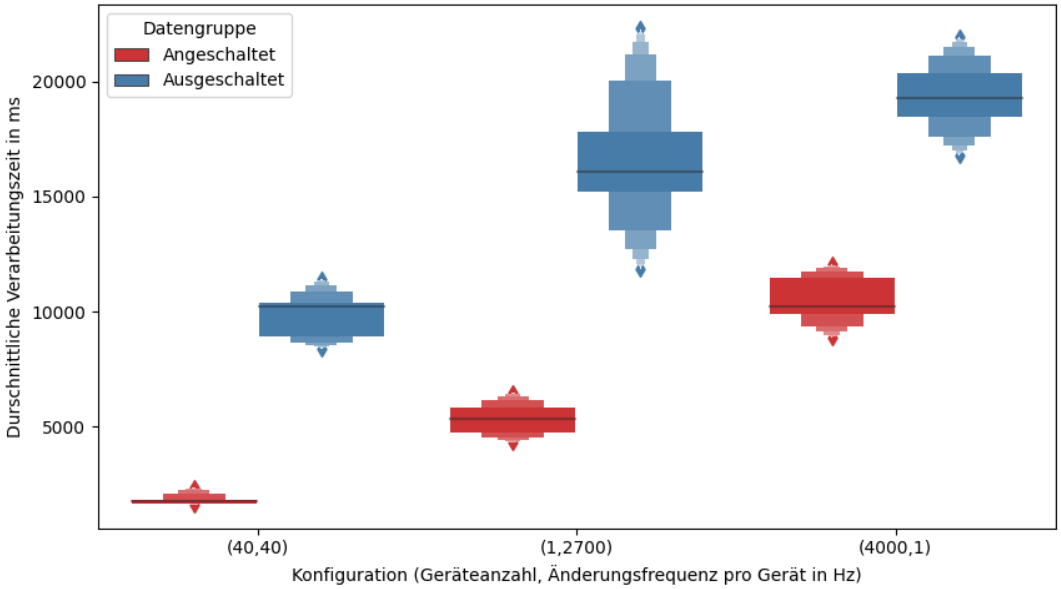

Abbildung 8.6: Verarbeitungszeiten mit und ohne Lese-Faktor in einer Datengruppe

8.2.1 Berechnung der Einsparung

Für die Evaluation der Speichereffizienz wird eine Berechnung hinzugezogen, welche tatsächliche Einsparungen für den Kanal eines Gerätes darlegt. Dabei müssen zunächst die Datentypen der zu speichernden Attribute sowie deren Größe bestimmt werden. Folgende Annahmen werden getroffen:

Attributname	Datentyp	Größe im Speicher
Geräte-ID	String	20 Byte
Zustand	String	20 Byte
Zeit der Aufnahme	Int64	8 Byte
Größe einzelner Eintrag gesamt		88 Byte

Weiterhin gilt für die zusätzlichen Attribute eines zusammengefassten Eintrags folgendes:

Schreib-Frequenz	Int32	4 Byte
Anzahl der zusammengefassten Daten	Int32	4 Byte
Größe zusammengefasster Eintrag gesamt		96 Byte

Die Formel für die Berechnung der Speichergröße, welche pro Geräte-Kanal mit entsprechender Schreibfrequenz (in der Datengruppe festgelegt) über eine bestimmte Dauer gespart wird, setzt sich wie in 8.1 beschrieben zusammen.

$$D = (f_E * D_E - f_S * D_Z) * t \tag{8.1}$$

mit

f_E = Frequenz erhaltener Daten
f_S = Schreibfrequenz
t = Dauer der Anwendung
D_E = Größe einzelner Eintrag
f_Z = Größe zusammengefasster Eintrag
D = Einsparung

Daraus ergeben sich beispielhaft folgende Einsparungen im Speicher für entsprechende Dauern.

Nach 1 Stunde (Einsparung in MB)

Frequenz erhaltener Daten \ Schreibfrequenz in Hz	1	10	100	1000
1	0	0	0	0
10	2,7	0	0	0
100	29,9	26,9	0	0
1000	301,0	299,0	269,0	0

Nach 24 Stunden (Einsparung in MB)

Frequenz erhaltener Daten \ Schreibfrequenz in Hz	1	10	100	1000
1	0	0	0	0
10	26,9	0	0	0
100	298,8	296,2	0	0
1000	3017,9	2988,2	2691,0	0

Nach 168 Stunden (1 Woche) (Einsparung in GB)

Frequenz erhaltener Daten \ Schreibfrequenz in Hz	1	10	100	1000
1	0	0	0	0
10	0,4	0	0	0
100	4,9	4,4	0	0
1000	49,5	49,0	44,2	0

Die Anwendung dieses Verfahrens eignet sich besonders in Umgebungen, in denen vor allem die Datenrate pro Gerät hoch ist, da nur pro Kanal Daten aggregiert werden.

8.2.2 Einfluss auf Performance

Nun soll erforscht werden, wie sich das Aktivieren und Deaktivieren des parallelen Threads zum Zusammenfassen entsprechend der Datengruppen auf die Performance der Eventverarbeitung auswirkt. Eine Einschränkung der Performance hinsichtlich der Eventverarbeitung ist den Abbildungen 8.7 und 8.8 deutlich zu erkennen. Ist der Prozess bei Ausführung des Programms aktiviert, sind höhere Verarbeitungszeiten sowie niedrigere Verarbeitungsfrequenzen messbar. Dies hat vermutlich den Grund, dass der Vorgang des Zusammenfassens noch nicht optimal an parallele Lasten des Systems eingeplant wird und stattdessen periodisch ausgeführt wird. Für die Integration des Mechanismus in das System bedarf es weiterer Verbesserungen, um trotz hoher Lasten eine maximale Performance zu erzielen.

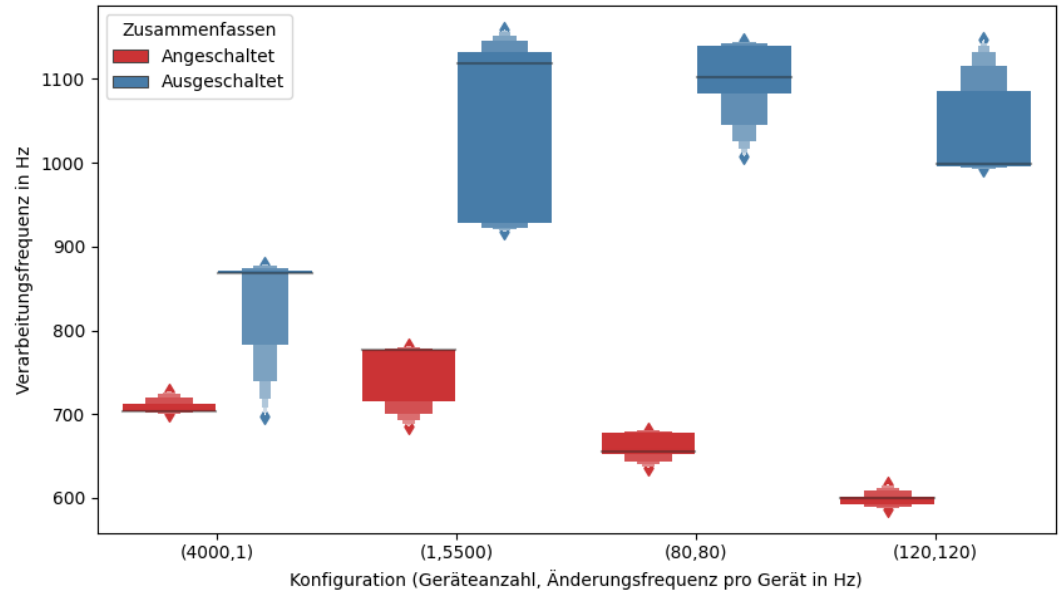

Abbildung 8.7: Einfluss des Zusammenfassens auf Verarbeitungsfrequenz

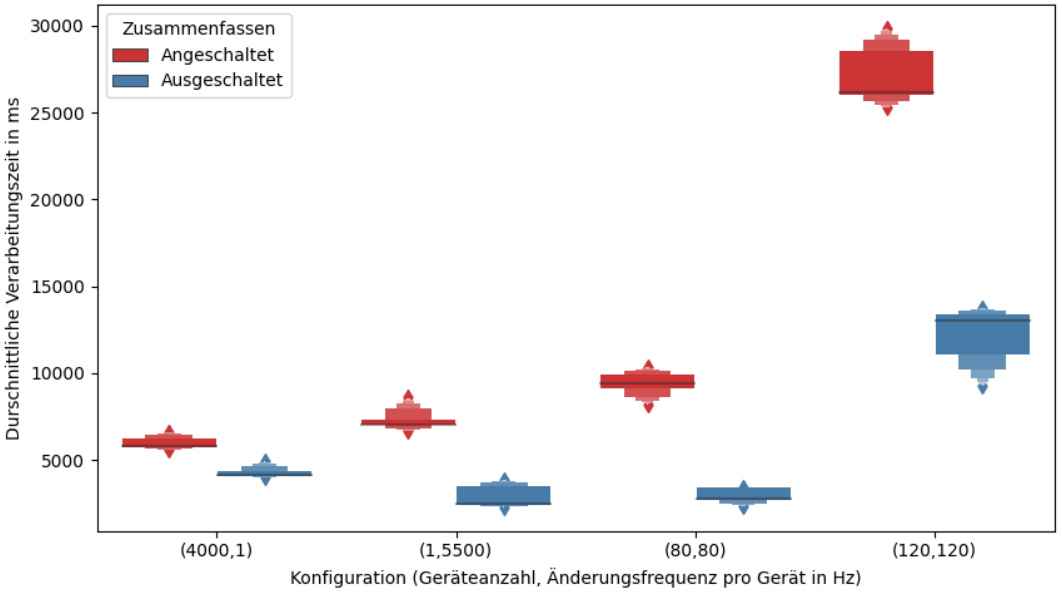

Abbildung 8.8: Einfluss des Zusammenfassens auf Verarbeitungszeit

9 Fazit

Das erfolgreiche Ausarbeiten eines Benchmark-Werkzeugs und dessen Implementation hat im Wesentlichen dazu beigetragen, die vorgestellten Systeme OpenHAB und Eraser, aber auch das eigens realisierte System Shunoled unter die Lupe zu nehmen und dabei die Performance zu messen. Dabei konnten unterschiedliche Ergebnisse festgestellt werden. Diese belegen, dass OpenHAB den Fokus nicht auf das Verarbeiten hoher Datenraten legt, sondern auf eine benutzerfreundliche und sehr erweiterbare Anwendung, welche eine komplexe und universelle Konfiguration für den Prozess der Automatisierung ermöglicht. Die Verarbeitungsfrequenz blieb unter 600 Hz, selbst bei höheren effektiven Gesamtsendefrequenzen. Hingegen erreicht Eraser vergleichsweise höhere Werte, welche bis zu einer Frequenz von 1500 Hz reichten.
Diese Messungen wurden auf Basis trivialer Konfigurationen der Regeln für die Automatisierung durchgeführt. Mit komplexeren Bedingungen in den Regeln und einer Anpassung der effektiven Gesamtsendefrequenz an die theoretische Gesamtsendefrequenz hätten die Ergebnisse sicherlich aussagekräftiger über die Verarbeitung von Events sein können. Letzteres bedarf einer umfassenden Konzipierung des Benchmark-Werkzeugs, in der das Absenden der Nachrichten zeitlich exakter eingeplant werden muss.
Die Ergebnisse der Gesamtperformance des umgesetzten Prototyps fielen etwas geringer aus als erwartet. Um diese zu verbessern, müssen verwendete Technologien und Mechanismen bedachter kombiniert werden. Dennoch sollte das Hauptaugenmerk auf der Evaluation der vorgeschlagenen Mechanismen liegen, da die Entwicklung des Prototyps als vollständiges SHS mehr Zeit in Anspruch nehmen würde, als für die Erstellung dieser Arbeit zur Verfügung stand. Eine Realisierung auf Basis von Sprachen wie C, C++ oder Rust würde länger andauern, allerdings höchstwahrscheinlich die Gesamtperformance maximieren, auch wenn dabei die Umsetzung von Erweiterungen erschwert wird.
Das Evaluieren der Realisierung von einzelnen Methodiken zur Erhöhung der Performance ergab, dass das Verwenden von den eingeführten Datengruppen mit einer Lesefrequenz zur Entlastung des Systems beitragen kann. Zustandsänderungen ausgehend von anderen Kanälen können dadurch mit höherer Frequenz verarbeitet werden. Zudem bewirkt das Empfangen von Duplikaten keine wesentliche Beeinträchtigung der Performance, wie das bei Eraser der Fall ist. Auch für das einhergehende Speicherproblem wurde eine Lösung mit akzeptablen Ergebnissen entwickelt. Das Zusammenfassen von hochfrequenten Daten basierend auf einer vom Nutzer gewählten Frequenz resultiert in einer kleineren Datenmenge, die abgespeichert werden muss. Diese konnte allerdings nur theoretisch berechnet und mit Beispielen gezeigt werden. Aufgrund der Einbußen in der Performance bei dem parallelen Prozess des Zusammenfassens müssen hier noch effiziente Techniken geschaffen werden, die das Zusammenspiel optimieren.
Letztlich stellen die kreierten Artefakte Ansätze für die dargelegten Herausforderungen und Probleme dar und können einen Beitrag zur Lösung eines modernen und an die heutige Zeit angepassten Smart Home Systems leisten.

Für allgemeine Verarbeitungen von Events bieten auch externe Arbeiten Ansätze, bei welchen der Fokus ebenfalls auf der Maximierung der Geschwindigkeit liegt. Darunter wurden Lösungen auf Basis von Ontologien oder einer verteilten Architektur vorgestellt. Diese basieren auf CEP und kommen hauptsächlich in der Industrie zur Anwendung. Dort gelten aufgrund des Einsatzes vernetzter technischer Elemente und komplexer Ereignisse ähnliche Bedingungen, wodurch gemeinsame Ziele gelöst werden müssen. Daher wird das Entwickeln einer abstrakten Software zur Eventverarbeitung als äußerst sinnvoll erachtet. Diese kann im Anschluss in die entsprechenden Domänen integriert werden.
Die aktuelle Kollaboration namhafter Unternehmen wie Google, Amazon und Apple, welche die Schaffung eines neuen, einheitlichen Verbindungsstandards für den Smart Home Sektor bezweckt, könnte neben der Einführung einer universellen Schnittstelle auch eine schlankere Software für Smart Home Systeme mit sich bringen, wodurch ebenfalls positive Auswirkungen auf die Performance zukünftiger Systeme zu erwarten sein dürften. Zudem sollen dabei markterprobte Technologien der genannten Unternehmen genutzt werden.

Abbildungsverzeichnis

Literatur

[1] *2017: THE YEAR OF THE SMART HOME*. URL: https://www.vectorsecurity.com/userfiles/file/pdf/blog/vs_smart-home-infographic_v1.pdf (besucht am 06. 08. 2020).

[2] *A laser pointer could hack your voice-controlled virtual assistant*. University of Michigan News. Section: News Releases. 5. Nov. 2019. URL: https://news.umich.edu/a-laser-pointer-could-hack-your-voice-controlled-virtual-assistant/ (besucht am 23. 08. 2020).

[3] Infineon Technologies AG. *Was ist ein Smart Home? - Infineon Technologies*. URL: https://www.infineon.com/cms/de/discoveries/smart-home-basics/ (besucht am 26. 07. 2020).

[4] KS Agarwal Sarthak; Rajan. "Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries". In: (2017). Publisher: University of Massachusetts Amherst. DOI: 10.7275/R5736P26. URL: http://scholarworks.umass.edu/foss4g/vol17/iss1/4/ (besucht am 02. 08. 2020).

[5] *Amazon, Apple, Google, die Zigbee Alliance und deren Vorstandsmitglieder bilden eine Arbeitsgruppe zur Entwicklung eines offenen Standards für Smart Home-Geräte*. Apple Newsroom. URL: https://www.apple.com/de/newsroom/2019/12/amazon-apple-google-and-the-zigbee-alliance-to-develop-connectivity-standard/ (besucht am 23. 08. 2020).

[6] S. Boraan Aruoba u. a. *credit, including © notice, is given to the source. A Comparison of Programming Languages in Economics*. 2014.

[7] *Automation Examples - Home Assistant*. URL: https://www.home-assistant.io/docs/automation/examples/ (besucht am 23. 08. 2020).

[8] *Benchmarking popular MQTT + JSON implementations*. URL: https://flespi.com/blog/benchmarking-popular-mqtt-json-implementations (besucht am 23. 08. 2020).

[9] Richa Bhatia. *NoSQL vs SQL — Which Database is Better For Big Data applications*. Analytics India Magazine. 6. Dez. 2017. URL: https://analyticsindiamag.com/nosql-vs-sql-database-type-better-big-data-applications/ (besucht am 23. 08. 2020).

[10] *binakot/Abstract-Rest-Service-Benchmark: A set of template services in different languages and frameworks. An abstract performance comparison based on the endpoint throughput. There are no objective benefits, just an elementary comparison of primitive RESTful API services.* URL: https://github.com/binakot/Abstract-Rest-Service-Benchmark (besucht am 26. 07. 2020).

[11] *Classification of Functions in Smart Home*. URL: https://www.researchgate.net/profile/Peter_Hamernik/publication/275156028_Classification_of_Functions_in_Smart_Home/links/55348d0a0cf2f2a588b25df8/Classification-of-Functions-in-Smart-Home.pdf (besucht am 02. 08. 2020).

[12] *Configuration*. URL: https://www.openhab.org/docs/configuration/ (besucht am 10. 08. 2020).

[13] Gianpaolo Cugola und Alessandro Margara. "Processing flows of information: From data stream to complex event processing". In: *ACM Computing Surveys* 44.3 (14. Juni 2012), 15:1–15:62. ISSN: 0360-0300. DOI: 10.1145/2187671.2187677. URL: https://doi.org/10.1145/2187671.2187677 (besucht am 23. 08. 2020).

[14] Pinchen Cui. "Comparison of IoT Application Layer Protocols". In: (), S. 63. URL: https://etd.auburn.edu/bitstream/handle/10415/5713/Pinchen_thesis.pdf?isAllowed=y&sequence=2 (besucht am 16. 08. 2020).

[15] *Cyberangriffe auf Yachten wirksam verhindern*. URL: https://www.sicherheit.info/cyberangriffe-auf-yachten-wirksam-verhindern (besucht am 29. 07. 2020).

[16] *Dataweave - CherryPy vs Sanic: Which Python API Framework is Faster?* URL: https://dataweave.com/blog/cherrypy-vs-sanic-which-python-api-framework-is-faster-103fe732adc6 (besucht am 23. 08. 2020).

[17] *Detailed access control and user management by reverse proxy - it works*. openHAB Community. 2. Feb. 2019. URL: https://community.openhab.org/t/detailed-access-control-and-user-management-by-reverse-proxy-it-works/66450 (besucht am 23. 08. 2020).

[18] *Die Ferienwohnung wird smart — E WIE EINFACH*. URL: https://www.e-wie-einfach.de/magazin/sicherheit/die-ferienwohnung-wird-smart (besucht am 30. 06. 2020).

[19] *Domoticz*. URL: https://www.domoticz.com/ (besucht am 23. 08. 2020).

[20] Michael Eckert und François Bry. "Complex Event Processing (CEP)". In: *Informatik-Spektrum* 32.2 (Apr. 2009), S. 163–167. ISSN: 0170-6012, 1432-122X. DOI: 10.1007/s00287-009-0329-6. URL: http://link.springer.com/10.1007/s00287-009-0329-6 (besucht am 15. 08. 2020).

[21] *Eigenschaften und Vorteile der Programmiersprache Python*. URL: http://www.michael-holzapfel.de/progs/python/python_vorteile.htm (besucht am 02. 08. 2020).

[22] Malin Eriksson und Victor Hallberg. *Supervisor: Mads Dam Examiner: Mads Dam*.

[23] *FHEM Gruppen anlegen*. AnWass Media. Section: FHEM. 26. März 2017. URL: https://anwass.de/fhem-gruppen-anlegen/ (besucht am 23. 08. 2020).

[24] *Home of FHEM*. URL: https://fhem.de/ (besucht am 23. 08. 2020).

[25] *Intelligente Hotelzimmer: Smart Home im Hilton*. URL: https://www.smart-wohnen.de/haus-garten/artikel/intelligente-hotelzimmer-smart-home-im-hilton/ (besucht am 23. 08. 2020).

[26] *Introduction to JSON — Berkeley Boot Camps — Data*. Berkeley Boot Camps. URL: https://bootcamp.berkeley.edu/resources/coding/learn-data-analytics/introduction-to-using-json-with-data-analytics/ (besucht am 28. 07. 2020).

[27] *ioBroker Smarthome*. URL: https://www.iobroker.net/ (besucht am 23. 08. 2020).

[28] *Items — openHAB*. URL: https://www.openhab.org/docs/configuration/items.html (besucht am 28. 07. 2020).

[29] Bill Karakostas. "A high performance engine for concurrent complex event processing". In: *Concurrency and Computation: Practice and Experience* 26.2 (2014). _eprint: https://onlinelibrary.wiley.com/do S. 491–499. ISSN: 1532-0634. DOI: 10.1002/cpe.3014. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3014 (besucht am 18. 08. 2020).

[30] Suman Kashyap u. a. *Benchmarking and Analysis of NoSQL Technologies*.

[31] Anthony Maeder und Patricia Williams. "Health Smart Homes: New Challenges". In: *MEDINFO 2017: Precision Healthcare through Informatics - Proceedings of the 16th World Congress on Medical and Health Informatics*. 16th World Congress of Medical and Health Informatics: Precision Healthcare through Informatics, MedInfo 2017. IMIA und IOS Press, 1. Jan. 2017, S. 166–169. DOI: 10.3233/978-1-61499-830-3-166. URL: https://researchnow.flinders.edu.au/en/publications/health-smart-homes-new-challenges (besucht am 26. 07. 2020).

[32] *Meet the Google Home app - Android - Chromecast Help*. URL: https://support.google.com/chromecast/answer/7071794?co=GENIE.Platform%3DAndroid&hl=en (besucht am 23. 08. 2020).

[33] Xianghang Mi u. a. "An empirical characterization of IFTTT: ecosystem, usage, and performance". In: *Proceedings of the 2017 Internet Measurement Conference*. IMC '17: Internet Measurement Conference. London United Kingdom: ACM, Nov. 2017, S. 398–404. ISBN: 978-1-4503-5118-8. DOI: 10.1145/3131365.3131369. URL: https://dl.acm.org/doi/10.1145/3131365.3131369 (besucht am 18. 08. 2020).

[34] *MQTT — CS181U Spring 2020*. URL: https://cs.pomona.edu/classes/po181u/docs/labs/lab4/ (besucht am 23. 08. 2020).

[35] *MSU research finds a new way to hack Siri and Google Assistant with ultrasonic waves*. MSUToday — Michigan State University. URL: https://msutoday.msu.edu/news/2020/msu-research-finds-a-new-way-to-hack-siri-and-google-assistant-with-ultrasonic-waves/ (besucht am 23. 08. 2020).

[36] *openHAB Add-ons*. URL: https://www.openhab.org/addons/ (besucht am 23. 08. 2020).

[37] *OpenHAB Architecture*. URL: https://upload.wikimedia.org/wikipedia/commons/b/bf/OpenHAB_Architecture.png (besucht am 23. 08. 2020).

[38] Rodrigo Ramirez. *Compiled vs Interpreted Code Performance*. Medium. 21. Okt. 2019. URL: https://medium.com/swlh/compiled-vs-interpreted-code-performance-e1a63299760b (besucht am 02. 08. 2020).

[39] *Rules*. URL: https://www.openhab.org/docs/configuration/rules-dsl.html (besucht am 15. 08. 2020).

[40] *Rules — openHAB*. URL: https://www.openhab.org/docs/configuration/rules-dsl.html#time-based-triggers (besucht am 15. 08. 2020).

[41] Omran Saleh. *Complex Event Processing in Wireless Sensor Networks*.

[42] René Schöne u. a. "Bridging the Gap between Smart Home Platforms and Machine Learning using Relational Reference Attribute Grammars". In: *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). Sep. 2019, S. 533–542. DOI: 10.1109/MODELS-C.2019.00083.

[43] *Securing Communication and Access*. URL: https://www.openhab.org/docs/installation/security.html (besucht am 23. 08. 2020).

[44] *Sitemaps — openHAB*. URL: https://www.openhab.org/docs/configuration/sitemaps.html#element-type-video (besucht am 25. 07. 2020).

[45] *Smart Device :: smart device :: ITWissen.info*. URL: https://www.itwissen.info/Smart-Device-smart-device.html (besucht am 15. 08. 2020).

[46] *Smart Home - worldwide — Statista Market Forecast*. Statista. URL: https://www.statista.com/outlook/283/100/smart-home/worldwide (besucht am 23. 08. 2020).

[47] *Smart Home-Health Care für Senioren - Smart Home-Geräte*. URL: https://smart-home-geraete.de/informationen-ueber-smart-home/smart-home-health-care-fuer-senioren/ (besucht am 23. 08. 2020).

[48] *Smart Homes Statistics – Fascinating Industry Trends*. 3D Insider. 11. Feb. 2020. URL: https://3dinsider.com/smart-home-statistics/ (besucht am 23. 08. 2020).

[49] *smart-door.net*. URL: https://www.smart-door.net/ (besucht am 29. 07. 2020).

[50] *So integrierst du Nuki in dein Smart Home System*. Nuki. URL: https://nuki.io/de/integrationen/ (besucht am 14. 08. 2020).

[51] Hagen Sparka u. a. "Effective Lossless Compression of Sensor Information in Manufacturing Industry". In: *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. 2017 IEEE 42nd Conference on Local Computer Networks (LCN). Singapore: IEEE, Okt. 2017, S. 480–488. ISBN: 978-1-5090-6523-3. DOI: 10.1109/LCN.2017.89. URL: http://ieeexplore.ieee.org/document/8109390/ (besucht am 18. 08. 2020).

[52] Cezary Szczegielniak u. a. *Tutors:* 2006. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.144.9670&rep=rep1&type=pdf (besucht am 23. 08. 2020).

[53] Kerry Taylor und Lucas Leidinger. "Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks". In: *The Semanic Web: Research and Applications*. Hrsg. von Grigoris Antoniou u. a. Bd. 6644. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 285–299. ISBN: 978-3-642-21063-1 978-3-642-21064-8. DOI: 10.1007/978-3-642-21064-8_20. URL: http://link.springer.com/10.1007/978-3-642-21064-8_20 (besucht am 15. 08. 2020).

[54] *Things*. URL: https://www.openhab.org/docs/configuration/things.html (besucht am 05. 08. 2020).

[55] *TOP Fast Python Web Frameworks in 2019 — Start Matter*. URL: https://blog.startmatter.com/top-fast-python-web-frameworks-in-2019/ (besucht am 26. 07. 2020).

[56] *Ubiquitious Computing*. URL: http://www.teco.edu/static/lehre/ubiqws0001old/ubiq2000-2/devices-grundlagen-8.pdf (besucht am 15. 08. 2020).

[57] Pieter Vromant u. a. "On interacting control loops in self-adaptive systems". In: *In: Proceedings of Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011*. 2011.

[58] Ruihan Wang und Zongyan Yang. "SQL vs NoSQL: A Performance Comparison". In: (), S. 3.

[59] *Was ist ein Smart Home? So funktioniert das intelligente Zuhause*. URL: https://g-pulse.de/was-ist-smart-home#keyfact-1 (besucht am 23. 08. 2020).

[60] *Zuhause und Geräte in der Google Home App teilen - Google Nest-Hilfe*. URL: https://support.google.com/googlenest/answer/9155535?hl=de (besucht am 30. 07. 2020).